VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

KRYPTOGRAFIE A OCHRANA SOUKROMÍ
CRYPTOGRAPHY AND PRIVACY PROTECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Ondrej Malík**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Hajný, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Ondrej Malík ***ID:*** 195818
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Kryptografie a ochrana soukromí

POKYNY PRO VYPRACOVÁNÍ:

Téma je zaměřeno na kryptografické metody pro ochranu soukromí v kyberprostoru. Výstupem práce je webová aplikace realizující protokoly na vydávání a ověřování osobních atributů dle schématu CDH16. Cílem práce je implementace funkcí pro vydání osobních atributů na čipovou kartu a realizace přívětivého uživatelského rozhraní. K aplikaci bude vytvořena kompletní dokumentace (uživatelská i instalační), vhodné uživatelské rozhraní, protokol z testování včetně měření základních parametrů (doba vydání atributů, zhodnocení kapacitních omezení, chybovost, atp.) a aplikace bude odladěna a optimalizována.

DOPORUČENÁ LITERATURA:

[1] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] MULTOS Developer's Guide [online]. , 96 [cit. 2019-09-06]. Dostupné z: https://www.multos.com/uploads/MDG.pdf

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: doc. Ing. Jan Hajný, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem této práce bylo vytvořit webové aplikace pro entity vydavatele, ověřovatele a revokační autority systému anonymní atributové autentizace RKVAC. Vytvořené aplikace poskytují veškeré funkce, které těmto entitám vyplývají z jejich podstaty. Aplikace tak vytvářejí komplexní fungující prostředí, v kterém je možná celková správa systému RKVAC. V aplikaci ověřovatele byl vytvořen autentizační modul, který umožňuje chránit webovou službu pomocí atributové autentizace RKVAC. Aplikace pro vydavatele a revokační autoritu jsou kompatibilní v rámci celého schématu RKVAC a je proto možná jejich implementace jako centrálních prvků systémů.

KLÍČOVÁ SLOVA

atributová autentizace, RKVAC, webová aplikace, řízení přístupu, zabezpečení webové služby, Node.js

ABSTRACT

The main goal of this diploma thesis was to create web applications for issuer, verifier and revocation authority of revocable keyed-verification anonymous credentials system. Applications created in this thesis provide functions for all tasks, that are performed by each entity. Using these applications a global management of RKVAC system is possible. Authentication module created in verifier's app is universaly usable for access control to any web service. Both issuer's and revocation authority's app are compatible with whole RKVAC system and are therefor applicable as central elements of systems.

KEYWORDS

attribute authentication, RKVAC, web application, access control, protecting web service, Node.js

MALÍK, Ondrej. *Kryptografie a ochrana soukromí*. Brno, 2021, 77 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jan Hajný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Kryptografie a ochrana soukromí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Janu Hajnému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

This thesis was created as part of the key activity KA6 - Individual teaching and involvement of students of bachelor's and master's degree programs in research within the project OP RDE Creating double-degree doctoral study program Electronics and Information Technology and creating doctoral study program Information Security reg. no. CZ.02.2.69/0.0/0.0/16_018/0002575.

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The project is co-financed by the European Union.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V posledních letech je na bezpečnost informačních technologií kladen stále větší důraz. S přesouváním každodenního života do kyberprostoru je nutné čím dál více dbát na ochranu soukromí uživatelů. Vývojáři technologií jsou ze strany veřejných autorit, ale i samotných uživatelů, nuceni do neustálého zvyšování bezpečnosti systémů. Zabezpečení dat a ochranu citlivých údajů řeší ve většině případů moderní kryptografie. V rámci kryptografie jsou vyvíjeny pokročilé bezpečnostní schémata, které přispívají k zvýšení ochrany citlivých dat během různých úkonů v kyberprostoru. Jedním z takových schémat je také schéma atributové autentizace CDH16, též označováno RKVAC (Revocable Keyed-Verification Anonymous Credentials).

Zadáním této práce je vytvořit webové aplikace, které budou sloužit pro jednotlivé entity schématu RKVAC. Aplikace budou poskytovat veškeré funkce, pro ovládání entit a budou vytvářet ucelený systém nástrojů pro správu systému RKVAC. Aplikace budou optimalizovány a vhodně odladěny. Součásti této práce je také poskytnout uživatelský a instalační manuál, spolu s důkazem fungování aplikací.

První část práce je zaměřena na teorii atributové autentizace a schématu RKVAC. Jsou zde popsány všechny strany vystupující v schématu a jeho veškeré protokoly. V druhé části je provedena analýza současného stavu vývoje aplikací pro schéma RKVAC. Výstupy analýzy jsou vybrané části vývoje, které budou dále použity jako základ pro webové aplikace vyvíjené v této práci. Třetí část práce pojednává o samotných vytvořených aplikacích. Jsou zde uvedeny principy, na kterých jsou aplikace založeny, popis grafického uživatelského rozhraní a také podrobnosti fungování aplikací z vývojářského hlediska. V přílohách práce je připojen návod k instalaci aplikací, uživatelský návod a důkaz fungování aplikací.

1 Teorie atributové autentizace

Základní funkcí autentizace je prokázání své identity. V digitálním světe nastávají situace, kdy v rámci autentizačního procesu není potřeba vyzradit celou identitu uživatele, ale jenom prokázat vlastnictví určitého osobního atributu. Takovým atributem může být například věk, národnost, nebo datum narození uživatele. V celém procesu je snaha chránit citlivé údaje uživatele v nejvyšší možné míre. Z tohoto důvodu jsou často využívána schémata založená na atributové autentizaci.

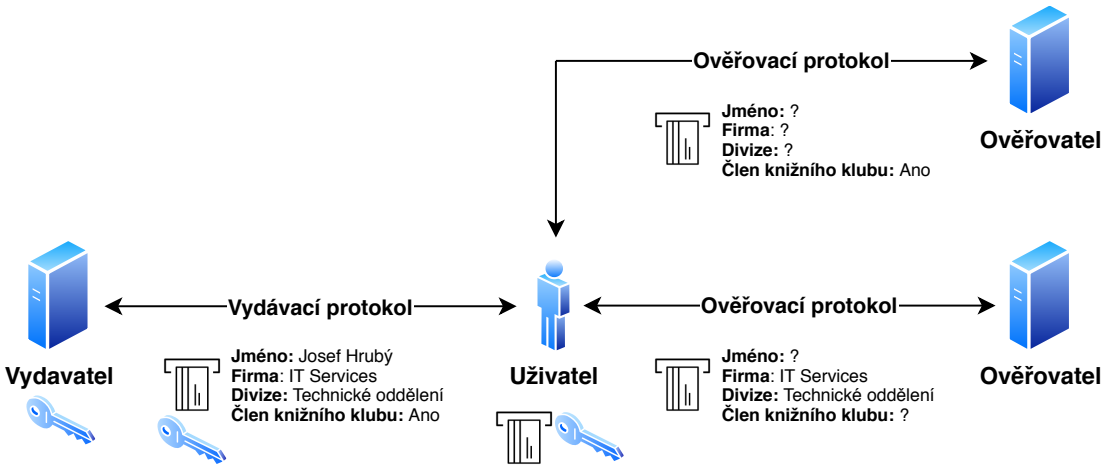
Atributová autentizace je autentizační proces, při kterém dochází k vyzrazení jenom nutných informací, s cílem ochránit identitu uživatele. Důsledkem je vyšší bezpečnost citlivých údajů před jejich zneužitím. Protokoly atributové autentizace jsou v souladu s evropskými právními předpisy o ochraně osobních údajů, např. GDPR [1].

Základní snahou atributové autentizace je utajení co nejvíce citlivých údajů během celého procesu. To nám dává možnost využívat bezpečnou autentizaci i v situacích, kdy chceme zachovat anonymitu. Typickým příkladem systému vhodného pro atributovou autentizaci může být systém řízení přístupu ve firmě. Pro lepší pochopení fungování atributové autentizace v praxi si popíšeme následnou modelovou situaci: každý zaměstnanec fiktivní firmy *IT Services s. r. o.* vlastní čipovou kartu s osobními údaji. Při vstupu do budovy přiložením karty k terminálu prokáže, že je zde zaměstnaný (je vlastníkem atributu *zaměstnanec IT Services*) a systém mu poté umožní vstup. Při vstupu do části budovy, kde jsou umístěny servery, musí systému prokázat, že je členem technického oddělení a má proto povolení vstoupit (vlastnictví atributu *technické oddělení*). V případě že si bude chtít zapůjčit knihu z firemní knihovny, musí prokázat, že je členem firemního knižního klubu. Autentizační schéma uvedeného příkladu je možné vidět na Obr. 1.1. Během procesu autentizace jsou systému poskytnuté vždy jenom atributy nutné k ověření dané informace. Ostatní osobní údaje jsou utajeny, spolu s identitou uživatele.

Aby byl protokol atributové autentizace považován za bezpečný, musí být **nespojitelný** a **nesledovatelný**. V praxi to znamená, že ověřovatelé nejsou schopni přiřadit jednotlivé dokazovací relace jednomu uživateli a taktéž na základě jednotlivých získaných atributů nedokáží odhalit identitu uživatele. Protokol by měl také umožňovat **selektivní uvolnění atributů**, díky čemuž si uživatel vždy může vybrat, které atributy poskytne ověřovateli.

1.1 Entity atributové autentizace

Jak je patrné z Obr. 1.1, schéma atributové autentizace obsahuje následující tři entity:



Obr. 1.1: Příklad procesu atributové autentizace

1. **vydavatel** – působí jako veřejná autorita a vydává atributy uživatelům,
2. **uživatel** – je vlastníkem zařízení, pomocí kterého prokazuje vlastnictví jednotlivých atributů,
3. **ověřovatel** – entita která ověřuje, že uživatel je vlastníkem určených atributů.

Některé systémy obsahují navíc i **revokační autoritu**. Její úlohou je případné odebrání vlastnictví atributů danému uživateli. V některých systémech je také schopná odhalit identitu uživatele (napr. schéma RKVAC, které bude implementováno v rámci této práce).

1.2 Kryptografické metody a primitiva

V této části budou podrobněji rozepsány kryptografické funkce a protokoly, které jsou používané v atributové autentizaci. Jejich vysvětlení je pro zvýšení přehlednosti doplněno o vhodné schémata.

1.2.1 Zero-knowledge a Σ-protokoly

Problémem autentizace založené na běžných kryptografických schématech (autentizace heslem, párem klíčů, certifikáty) je schopnost ověřovatele zjistit informace o uživateli, nebo dokonce odhalit jeho identitu. Zero-knowledge protokoly řeší tento problém tím, že umožňují uživateli dokázat znalost tajemství, bez zveřejnění jiných citlivých informací [2].

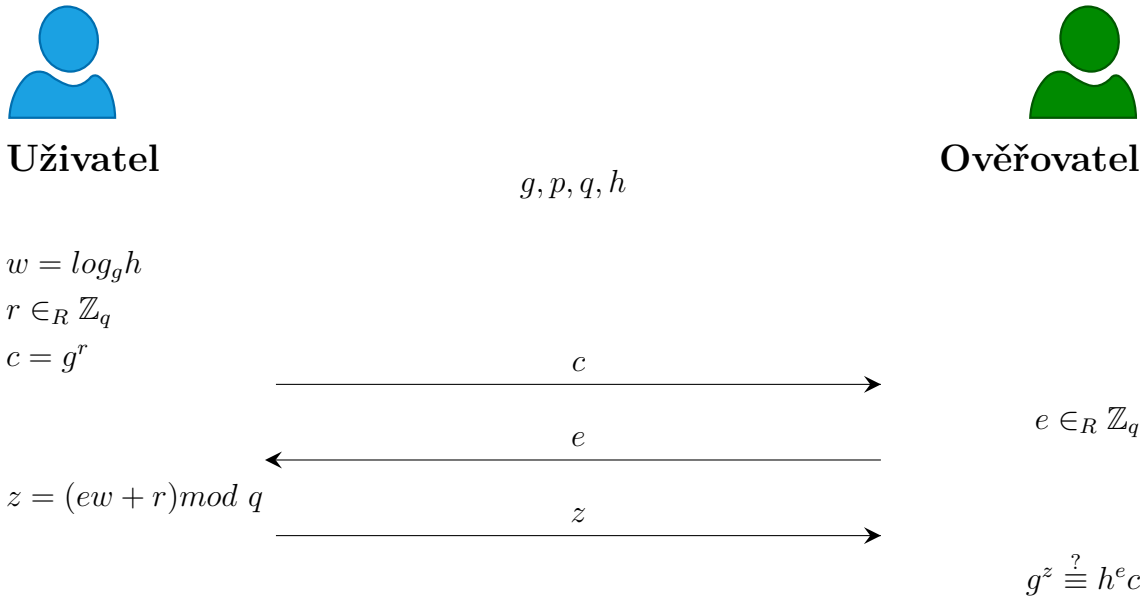
Zero-knowledge protokoly jsou příkladem interaktivních důkazových systémů, kde si uživatel a ověřovatel vyměňují více zpráv. Uživatel se snaží ověřovateli poskytnou důkaz znalosti tajemství w. Ověřovatel může důkaz odmítnout, nebo ho akceptovat. Tento důkaz však nesmí žádným způsobem odhalit tajemství w [2].

Aby byl protokol považován za zero-knowledge, musí splňovat následující vlastnosti [2]:

- **Úplnost** – protokol je považován za úplný v případě, kdy ověření mezi uživatelem, který zná tajemství w, a validním ověřovatelem skončí úspěchem.
- **Spolehlivost** – jestli uživatel který nezná tajemství w, není schopen ověřovateli dokázat, že ho zná, protokol je považován za spolehlivý.
- **Zero-knowledge** – protokol má tuto vlastnost, jestli během procesu není zveřejněna o uživateli jiná informace, než jeho znalost tajemství w.

Sigma protokoly (Σ-protokoly) jsou praktickou implementací zero-knowledge protokolů. Musí navíc splňovat vlastnost **třícestnost**, díky které jsou mnohem jednoduší na implementaci. Prvně posílá uživatel správu *commitment*, kterou se zavazuje k znalosti náhodné hodnoty. Následně přichází výzva od ověřovatele, na kterou uživatel reaguje posláním důkazu znalosti daného atributu [3].

Často používaným Σ-protokolem je Schnorrův protokol viz Obr. 1.2. Uživatel dokazuje znalost tajemství w pomocí důkazu znalosti diskrétního logaritmu. Problém diskrétního logaritmu vychází z neschopnosti vypočítat inverzním postupem exponent, který byl vstupem modulárního mocnění s velkým prvočíslem. Pomocí protokolu dokáže uživatel dokázat znalost tajemství $PK(w) : h = g^w mod\ p$, bez vyzrazení citlivých údajů. Uživatel, který tajemství zná, je vždy schopen dokázat jeho znalost a ověřovatel je schopen důkaz ověřit [4] pomocí vztahu: $g^z \equiv h^e c$.



Obr. 1.2: Schnorrův protokol

1.2.2 Podpisové schéma Weak Boneh-Boyen

V implementovaném schématu atributové autentizace obdrží uživatel od vydavatele podpis, založený na podpisovém schématu *Weak Boneh-Boyen*. Toto schéma umožňuje podpis randomizovat. Tím je původní podpis pozměněn způsobem, který zachovává jeho původní validitu, avšak s původním podpisem je nespojitelný. V praxi je pak možné ověřovateli prokázat znalost původního podpisu, bez jeho poskytnutí ověřovateli. Je tak zachována nespojitelnost, nesledovatelnost a anonymita.

Podpisové schéma v základním módu používá 3 algoritmy – `Setup`, `Sign` a `Verify`. Pro dokázání znalosti podpisu bez jeho zveřejnění jsou do schématu přidány algoritmy `Prove` a `VerifyRandomized`. Důležité metody pro implementované schéma jsou blíže popsané v následujícím výpisu [5]:

$(params, sk, pk) \leftarrow \Sigma.\texttt{Setup}(l_q)$:

- vstupem metody je bezpečnostní parametr l_q, pomocí kterého jsou určeny systémové parametry $params(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, \mathrm{e})$.
- výstupem je soukromý klíč $sk \in_R \mathbb{Z}_q$ a veřejný klíč $pk = g_2^{sk}$.

$(\sigma) \leftarrow \Sigma.\texttt{Sign}(sk, m)$:

- metoda spočítá podpis $\sigma = g_1^{\frac{1}{sk+m}}$ na základě zprávy m a soukromého klíče sk.

$(\hat{\sigma}, \overline{\sigma}, \pi) \leftarrow \Sigma.\texttt{Prove}(\sigma, m)$:

- pomocí tohoto algoritmu prokáže uživatel ověřovateli vlastnictví podpisu. Pomocí náhodného čísla $r \in_R \mathbb{Z}_q$ vypočte nový podpis $\hat{\sigma} = \sigma^m$ a pomocnou hodnotu $\overline{\sigma} = \hat{\sigma}^{-m}\ g_1^r$.
- algoritmus vrací důkaz o vlastnictví podpisu π.

$(0/1) \leftarrow \Sigma.\texttt{VerifyRandomized}(\hat{\sigma}, \overline{\sigma}, \pi, pk)$:

- ověřovatel vypočítá hodnotu $t' = g_1^{s_r} \sigma_r^{s_m - e*pk}$ a ověří znalost podpisu $\mathrm{e} \stackrel{?}{=} H(\hat{\sigma}, t', nonce)$.

1.2.3 Algebraický MAC s využitím wBB podpisu

Algebraický MAC (Message Authentication Code) je symetrický algoritmus sloužící na zajištění autentičnosti a integrity bloku zpráv. Je založen na *Weak Boneh-Boyen* podpisu popsaném v části 1.2.2. Stejně jako podpisové schéma `wBB` i algoritmus $\texttt{MAC}_{\texttt{wBB}}$ pozůstává z třech základních algoritmů `Setup`, `Sign` a `Verify`. Zároveň pro prokazatelnost znalosti autentizačního kódu MAC slouží přídavné metody `Prove` a `VerifyRandomized`.

$(params, sk) \leftarrow \texttt{MAC}_{\texttt{wBB}}.\texttt{Setup}(l_q)$:

- vstupem metody je bezpečnostní parametr l_q, pomocí kterého jsou určeny systémové parametry $params(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, \mathrm{e})$.
- výstupem jsou soukromé klíče $sk = (x_0, x_1, \ldots, x_n) \in_R \mathbb{Z}_q$.

$(\sigma, (\sigma_{x_0}, \ldots, \sigma_{x_n})) \leftarrow \texttt{MAC}_{\texttt{wBB}}.\texttt{Sign}(sk, (m_1, \ldots, m_n))$:

- metoda spočítá podpis $\sigma = g_1^{\frac{1}{x_0 + m_1 x_1 + \cdots + m_n x_n}}$ na základě množiny zpráv $(m_1, \ldots, m_n) \in \mathbb{Z}_q$ a soukromého klíče sk.
- zároveň jsou vypočteny pomocné hodnoty $(\sigma_{x_0}, \ldots, \sigma_{x_n}) = (\sigma^{x^0}, \ldots, \sigma^{x_n})$

$(0/1) \leftarrow \texttt{MAC}_{\texttt{wBB}}.\texttt{Verify}(sk, (m_1, \ldots, m_n), \sigma, (\sigma_{x_0}, \ldots, \sigma_{x_n}))$:

- v případě, že platí $g = \sigma^{x_0 + m_1 x_1 + \cdots + m_n x_n}$, algoritmus vyhodnotí podpis σ jako platný.

$(\hat{\sigma}, (\hat{\sigma}_{x_0}, \ldots, \hat{\sigma}_{x_n}), t) \leftarrow \texttt{MAC}_{\texttt{wBB}}.\texttt{Prove}(\sigma, (\sigma_{x_0}, \ldots, \sigma_{x_n}), (m_1, \ldots, m_n))$:

- pomocí tohoto algoritmu prokáže uživatel ověřovateli znalost autentizačního kódu. Pomocí náhodného čísla $r \in \mathbb{Z}_q$ vypočítá randomizovaný podpis $\hat{\sigma} = \sigma^r$ a pomocné hodnoty $(\hat{\sigma}_{x_0}, \ldots, \hat{\sigma}_{x_n}) = (\sigma_{x_0}^r, \ldots, \sigma_{x_n}^r)$.

$(0/1) \leftarrow \texttt{MAC}_{\texttt{wBB}}.\texttt{VerifyRandomize}(\hat{\sigma}, (\hat{\sigma}_{x_0}, \ldots, \hat{\sigma}_{x_n}), t, (m_1, \ldots, m_n))$:

- ověřovatel vypočítá hodnotu $t' = g_1^{s_r} \sigma_r^{s_m - e * x_0}$ a ověří znalost kódu MAC $\mathrm{e} \stackrel{?}{=} \mathcal{H}(\hat{\sigma}, t', nonce)$.

1.3 Revokovatelné schéma RKVAC

Schéma, které bude implementováno v rámci diplomové práce, je Revocable Keyed-Verification Anonymous Credentials. Jedná se o schéma atributové autentizace, které zabezpečuje anonymitu uživatele při ověřování znalosti atributů a splňuje všechny ostatní požadované vlastnosti, popsané v úvodu kapitoly 1. Schéma kromě tří základních entit – **vydavatel**, **uživatel** a **ověřovatel**, obsahuje entitu **revokační autorita**. Ta zabezpečuje revokovatelnost uživatele ze systému a jeho případnou deanonymizaci.

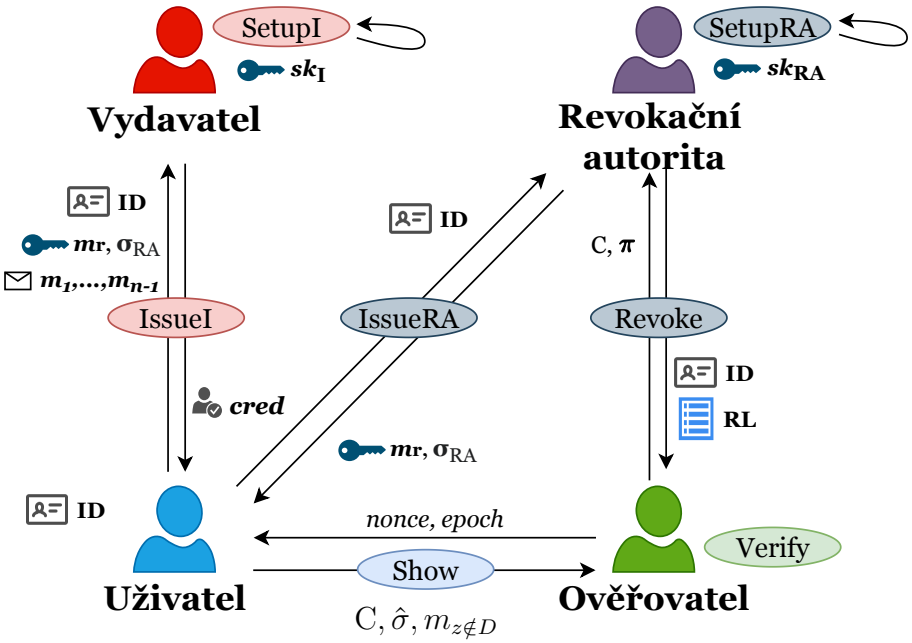
Jednotlivé entity v rámci schématu RKVAC mezi sebou komunikují pomocí následujících čtyř protokolů. Fungování jednotlivých protokolů je podrobněji popsáno v následujících částech.

- **Nastavení** – tento protokol obsahuje algoritmy `SetupI` a `SetupRA`, prováděné vydavatelem a revokační autoritou. Protokol se stará o generování parametrů

systému, nastavení soukromých a veřejných klíčů a jejich distribuci.

- **Vydávací protokol** – algoritmy `IssueI` a `IssueRA` slouží na vydávání a verifikaci osobních atributů uživateli. Protokol je vykonáván vydavatelem a revokační autoritou.
- **Ověřovací protokol** – pomocí algoritmů `Show` a `Verify` prokazuje uživatel ověřovateli držení osobních atributů.
- **Revokační protokol** – revokační autorita je schopna pomocí algoritmu `Revoke` zneplatnit držení atributů uživatelem a odhalit jeho identitu.

Zjednodušené fungování celého schématu RKVAC můžeme vidět na Obr. 1.3.



Obr. 1.3: Schéma atributové autentizace RKVAC

1.3.1 Nastavení

Na začátku fungování schématu je nutné nastavit entitám vydavatele a revokační autority požadované parametry. To se provede pomocí algoritmů `SetupI` a `SetupRA`.

$(params, sk) \leftarrow$ `SetupI`(l_q):

- vstupem metody je bezpečnostní parametr l_q, pomocí kterého jsou určeny systémové parametry $params(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathrm{e})$. Parametry $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ jsou cyklické grupy stejného řádu, g_1 a g_2 jsou body eliptické křivky řádu q a e označuje nedegenerativní mapu pro bilineární párování $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

- tento protokol je prováděn vydavatelem a jeho výstupem je jeho soukromý klíč $sk = (x_0, x_1, \ldots, x_n) \in_R \mathbb{Z}_q$.

$(params_{\mathtt{RA}}, sk_{\mathtt{RA}}, pk_{\mathtt{RA}}) \leftarrow \mathtt{SetupRA}(1^K, ver_{\max})$:

- vstupem tohto protokolu je parametr $ver_{\max}$, který určuje maximální počet ověřovacích relací za jednu časovou epochu. Díky tomu je zabezpečena nespojitelnost algoritmu. Hodnota je dána vztahem $ver_{\max} = k^j$, kde j určuje počet náhodných celočíselných proměnných α a k určuje počet randomizérů e a jejich podpisů σ_e. Vhodným zvolením parametrů j a k jsou optimalizovány výpočetní a paměťové nároky algoritmu.
- jsou vybrány náhodné proměnné $(\alpha_1, \ldots, \alpha_j) \in_R \mathbb{Z}_q$ a podle vztahu $h_j = g_1^{\alpha_z}$ jsou vypočteny body $(h_1, \ldots, h_j)$.
- je zvolen soukromý klíč revokační autority $sk_{\mathrm{RA}} \in_R \mathbb{Z}_q$ a následně z něho vypočten veřejný klíč $pk_{\mathrm{RA}} = g_2^{sk_{\mathrm{RA}}}$.
- náhodné hodnoty $(e_1, \ldots, e_j) \in_R \mathbb{Z}_q$ jsou podepsány $\sigma_{e_z} \leftarrow g_1^{\frac{1}{e_z + sk_{\mathrm{RA}}}}$ a tím vznikne sada randomizérů a jejich podpisů $((e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k}))$
- kromě páru klíčů revokační autority jsou na výstupu algoritmu parametry $params_{\mathrm{RA}} = (q, \mathbb{G}_1, g_1, k, j, (h_1, \ldots, h_j), (\alpha_1, \ldots, \alpha_j), ((e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})))$

1.3.2 Vydávací protokol

Vydávací protokol se stará o vydání atributů uživateli a o nastavení jeho revokačního handleru. Celý protokol je zabezpečován dvěma algoritmy – `IssueRA` a `IssueI`. Jako první je spuštěn protokol mezi uživatel a revokační autoritou.

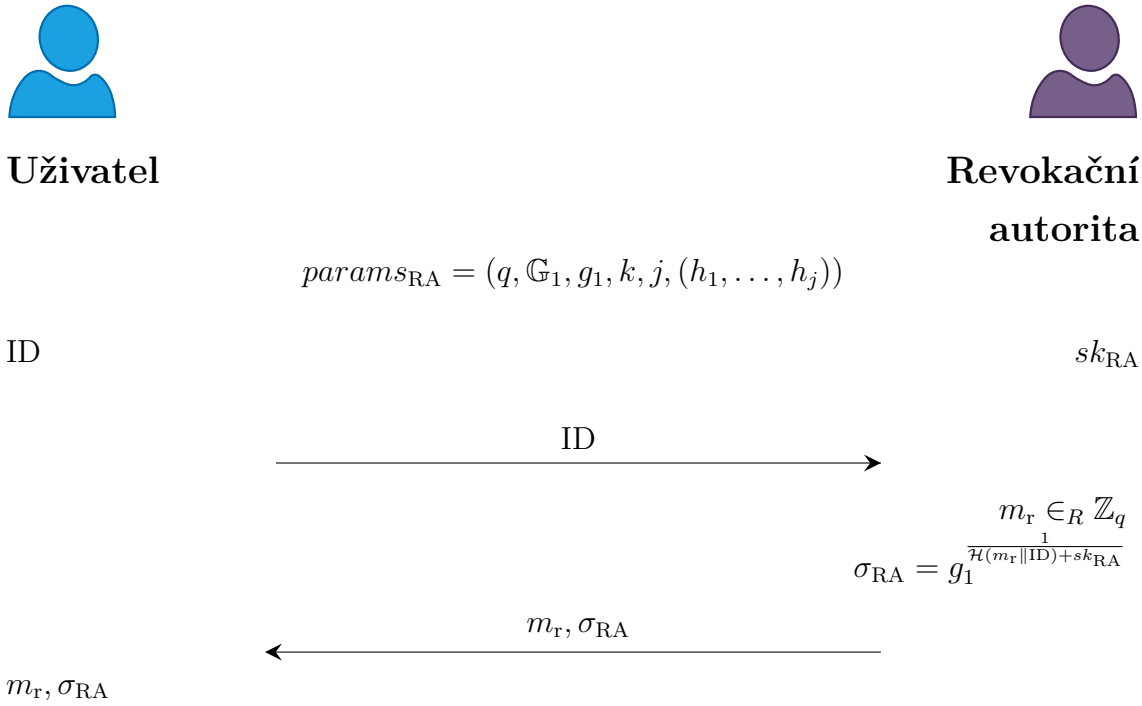
$(m_{\mathrm{r}}, \sigma_{\mathrm{RA}}) \leftarrow \mathtt{IssueRA}\,(params_{\mathrm{RA}}, sk_{\mathrm{RA}}, \mathrm{ID})$

- vstupem protokolu jsou parametry $params_{\mathrm{RA}}$, soukromý klíč sk_{RA} a identifikátor ID, který uživatel zašle revokační autoritě,
- revokační autorita generuje revokační handler m_{r} a vypočítá jeho podpis podle vztahu $\sigma_{\mathrm{RA}} = g_1^{\frac{1}{\mathcal{H}(m_{\mathrm{r}} \| \mathrm{ID}) + sk_{\mathrm{RA}}}}$
- revokační handler je spolu s pověřením poslán uživateli.

Algoritmus `IssueRA` je zobrazen na Obr. 1.4. Druhou části vydávacího protokolu je samotné vydání atributů vydavatelem uživateli.

$(cred) \leftarrow \mathtt{IssueI}(params_{\mathrm{I}}, sk_{\mathrm{I}}, pk_{\mathrm{RA}}, \mathrm{ID}, m_{\mathrm{r}}, \sigma_{\mathrm{RA}}, (m_1, \ldots, m_{n-1}))$:

- vstupem algoritmu jsou parametry $params_{\mathrm{I}}$, soukromý klíč vydavatele a veřejný klíč revokační autority,



Obr. 1.4: Algoritmus `IssueRA` vydávacího protokolu schématu RKVAC

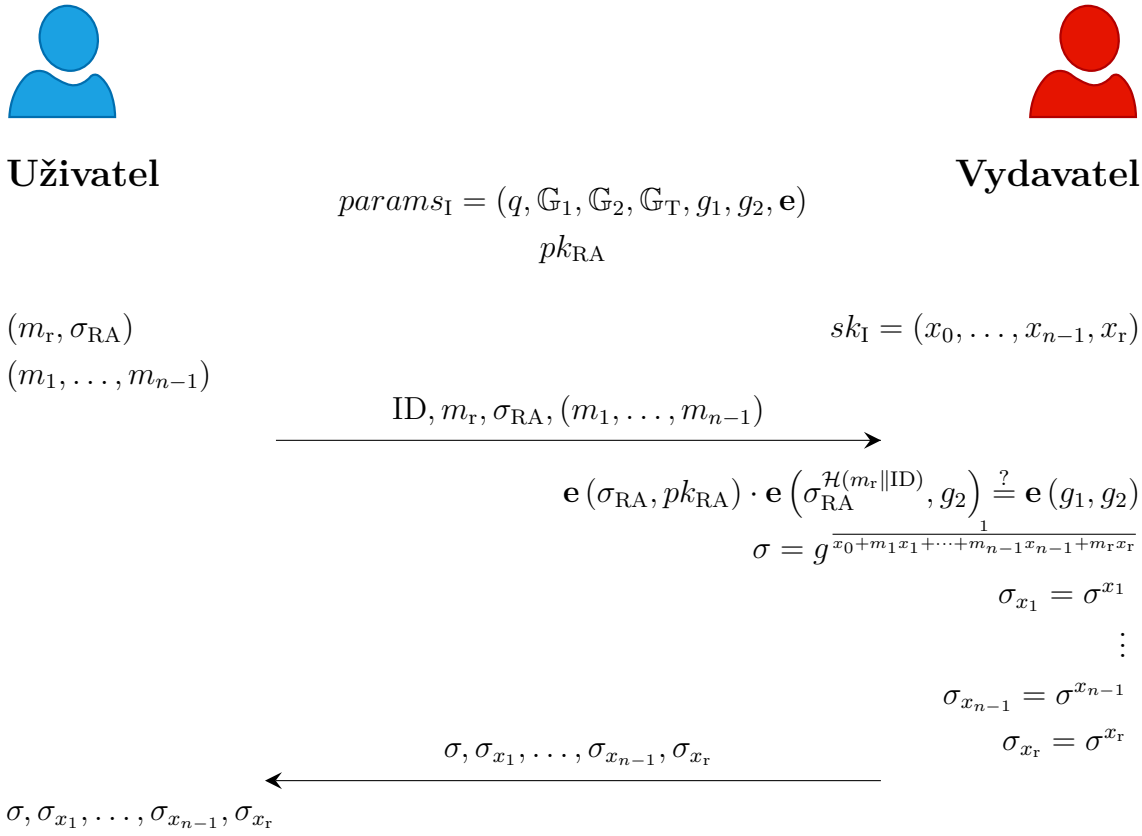
- uživatel pošle vydavateli své ID, revokační handler m_{r}, jeho podpis σ_{RA} a atributy, pro které chce získat podpis,
- vydavatel je povinen prověřit kredibilitu uživatele, aby tím zabránil podvrhnutí falešného revokačního handleře. To provede pomocí veřejného klíče revokační autority $\mathbf{e}(\sigma_{\mathrm{RA}}, pk_{\mathrm{RA}}) \cdot \mathbf{e}\left(\sigma_{\mathrm{RA}}^{\mathcal{H}(m_{\mathrm{r}} \| \mathrm{ID})}, g_2\right) \stackrel{?}{=} \mathbf{e}(g_1, g_2)$,
- v případě úspěšného ověření vydá uživateli pověření k atributům $cred = \left(\sigma, \sigma_{x_{\mathrm{r}}}, \sigma_{x_0}, \sigma_{x_1}, \ldots, \sigma_{x_{n-1}}\right)$.

Algoritmus `IssueI` můžeme vidět na Obr. 1.5.

1.3.3 Ověřovací protokol

Pomocí ověřovacího protokolu dokazuje uživatel ověřovateli znalost vybraných atributů. Je zcela na uživateli, které atributy ověřovateli odešle. Tím je zachována vlastnost **selektivního odhalení atributů**. Uživatel místo svého ID posílá ověřovateli pseudonym C. Pomocí toho je zachována **anonymita** uživatele a jeho **nesledovatelnost**. Pseudonym C je počítán také z náhodní hodnoty *nonce* a časové hodnoty *epoch*. Tím je zajištěna **nespojitelnost** jednotlivých ověřovacích relací.

Ověřovací protokol je zajištěn spoluprací algoritmu na prokázání držení atributů `Show` a algoritmu na následné ověření pověření ze strany ověřovatele `Verify`.



Obr. 1.5: Algoritmus `IssueI` vydávacího protokolu schématu RKVAC

$(\mathrm{C}, \hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}, \pi) \leftarrow$ `Show` $(params_{\mathrm{I,RA}}, m_{\mathrm{r}}, (m_1, \ldots, m_{n-1}), cred, epoch)$:

- uživatel od ověřovatele obdrží hodnoty *nonce* a *epoch*,
- pomocí jedinečné hodnoty i vypočtené z $params_{\mathrm{RA}}$ (randomizérů e_z a α_z viz sekce 1.3.1) $i = \sum_{z=1}^{j} \alpha_z e_z$, hashe $\mathcal{H}(epoch)$ a revokačního handleru m_{r} vypočte uživatel pseudonym C: $\mathrm{C} = g_1^{\frac{1}{i - m_{\mathrm{r}} + \mathcal{H}(epoch)}}$,
- uživatel generuje několik náhodných hodnot pro sestavení znáhodněného pověření $\hat{\sigma}$,
- následně je uživatelem sestaven důkaz znalosti π včetně důkazů znalosti pseudonymu: $(\hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \overline{\sigma}_{e_I}, \overline{\sigma}_{e_{II}}, \pi)$
- ověřovateli je poslán pseudonym, důkaz znalosti a znáhodněné pověření, včetně odhalených atributů pro ověření $m_{z \notin D}$.

$(0/1) \leftarrow$ `Verify` $(params_{\mathrm{I,RA}}, sk_{\mathrm{I}}, pk_{\mathrm{RA}}, m_{z \in D}, C, \hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}, \pi, epoch)$:

- ověřovatel ověřuje zdali je důkaz znalosti včetně důkazu znalosti pseudonymů platný a jestli se daný pseudonym nenachází na seznamu revokovaných uživatelů,

- v případě úspěšného ověření jsou zasláne atributy považovány za validní.

Celý průběh algoritmu je možné vidět na Obr. 1.6.

1.3.4 Revokační protokol

Na revokaci uživatel slouží algoritmus `Revoke`, který probíha mezi ověřovatelem a revokační autoritou. Ověřovatel dodá revokační autoritě transkript ověřovací transakce s uživatelem, kterého chce revokovat. Revokační autorita následně rozšíří revokační listinu o všechny pseudonymy C, které je daný uživatel schopen vypočítat. V praxi je tak uživatel revokován a není více schopen prokázat vlastnictví atributů.

Uživatel

Ověřovatel

$$params_{\mathrm{I}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}, g_1, g_2, \mathbf{e})$$
$$params_{\mathrm{RA}} = (q, \mathbb{G}_1, g_1, k, j, (h_1, \ldots, h_j)\,(\alpha_1, \ldots, \alpha_j))$$
$$pk_{\mathrm{RA}}, RL$$

$params_{\mathrm{RA}} = (e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})$ $\qquad$ $sk_{\mathrm{V}} = sk_{\mathrm{I}} = (x_0, \ldots, x_{n-1}, x_{\mathrm{r}})$

$(m_1, \ldots, m_{n-1}, m_{\mathrm{r}})$

$\sigma, \left(\sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_{\mathrm{r}}}\right)$

$nonce, epoch$ ←

$e_I, e_{II} \in_R (e_1, \ldots, e_k)$

$\sigma_{e_I}, \sigma_{e_{II}} \in_R (\sigma_{e_1}, \ldots, \sigma_{e_k})$

$i \leftarrow \alpha_1 e_I + \alpha_2 e_{II}$

$C \leftarrow g_1^{\frac{1}{i - m_{\mathrm{r}} + \mathcal{H}(epoch)}}$

$\rho, \rho_v, \rho_i, \rho_{m_{\mathrm{r}}}, \rho_{m_{z \notin D}}, \rho_{e_I}, \rho_{e_{II}} \in_R \mathbb{Z}_q$

$\hat{\sigma} \leftarrow \sigma^{\rho}$

$\hat{\sigma}_{e_I} \leftarrow \sigma_{e_I}^{\rho}, \hat{\sigma}_{e_{II}} \leftarrow \sigma_{e_{II}}^{\rho}$

$\bar{\sigma}_{e_I} \leftarrow \hat{\sigma}_{e_I}^{-e_I} g_1^{\rho}, \bar{\sigma}_{e_{II}} \leftarrow \hat{\sigma}_{e_{II}}^{-e_{II}} g_1^{\rho}$

$t_{\text{verify}} \leftarrow g_1^{\rho_v} \sigma^{\rho_{m_{\mathrm{r}}} \rho} \prod_{z \notin D} \sigma_{x_z}^{\rho_{m_z} \rho}$

$t_{\text{revoke}} \leftarrow C^{\rho_{m_{\mathrm{r}}}} C^{\rho_i}$

$t_{\text{sig}} \leftarrow g_1^{\rho_i} h_1^{\rho_{e_I}} h_2^{\rho_{e_{II}}}, t_{\text{sigI}} \leftarrow g_1^{\rho_v} \hat{\sigma}_{e_I}^{\rho_{e_I}}, t_{\text{sigII}} \leftarrow g_1^{\rho_v} \hat{\sigma}_{e_{II}}^{\rho_{e_{II}}}$

$e \leftarrow \mathcal{H}\left(t_{\text{verify}}, t_{\text{revoke}}, t_{\text{sig}}, t_{\text{sigI}}, t_{\text{sigII}}, \hat{\sigma}, \hat{\sigma}_{e_I}, \bar{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_{II}}, C, nonce\right)$

$\langle s_{m_z} \leftarrow \rho_{m_z} - e m_z \rangle_{z \notin D}$

$s_v \leftarrow \rho_v + e\rho$

$s_{m_{\mathrm{r}}} \leftarrow \rho_{m_{\mathrm{r}}} - e m_{\mathrm{r}}$

$s_i \leftarrow \rho_i + e i$

$s_{e_I} \leftarrow \rho_{e_I} - e e_I, s_{e_{II}} \leftarrow \rho_{e_{II}} - e e_{II}$

$\pi \leftarrow \left(e, s_{m_{z \notin D}}, s_v, s_{m_{\mathrm{r}}}, s_i, s_{e_I}, s_{e_{II}}\right)$

$m_{z \in D}, \pi, C, \hat{\sigma}, \hat{\sigma}_{e_I}, \bar{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_{II}}$ →

$t_{verify} \leftarrow \hat{\sigma}^{-e x_0} g_1^{s_v} \hat{\sigma}^{x_{\mathrm{r}} s_{m_{\mathrm{r}}}} \prod_{z \notin D} \hat{\sigma}^{x_z s_{mz}} \prod_{z \in D} \hat{\sigma}^{-e x_z s_{mz}}$

$t_{revoke} \leftarrow \left(g_1 C^{-\mathcal{H}(epoch)}\right)^{-e} C^{s_{m_{\mathrm{r}}}} C^{s_i}$

$t_{\text{sig}} \leftarrow g_1^{s_i} h_1^{s_{e_I}} h_2^{s_{e_{II}}}, t_{\text{sigI}} \leftarrow g_1^{s_v} \hat{\sigma}_{e_I}^{s_{e_I}} \bar{\sigma}_{e_I}^{-e}, t_{\text{sigII}} \leftarrow g_1^{s_v} \hat{\sigma}_{e_{II}}^{s_{e_{II}}} \bar{\sigma}_{e_{II}}^{-e}$

$e \leftarrow \mathcal{H}\left(t_{\text{verify}}, t_{\text{revoke}}, t_{\text{sig}}, t_{\text{sigI}}, t_{\text{sigII}}, \hat{\sigma}, \hat{\sigma}_{e_I}, \bar{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_{II}}, C, nonce\right)$

$\mathrm{e}\left(\bar{\sigma}_{e_I}, g_2\right) = \mathrm{e}\left(\hat{\sigma}_{e_I}, pk_{\mathrm{RA}}\right)$

$\mathrm{e}\left(\bar{\sigma}_{e_{II}}, g_2\right) = \mathrm{e}\left(\hat{\sigma}_{e_{II}}, pk_{\mathrm{RA}}\right)$

$C \notin RL$

Obr. 1.6: Algoritmy `Show` a `Verify` ověřovacího protokolu schématu RKVAC

2 Analýza současného stavu

Druhá kapitola práce je zaměřená na analýzu aktuálních implementací schématu RKVAC. Pojednává o třech bakalářských pracích, vytvořených v akademickém roce 2019/2020. Jedná se o práce [7], [8] a [9]. Všechny práce implementovaly nějakou část schématu RKVAC.

Práce [7] se zabývá vývojem aplikace pro programovatelné čipové karty na platformě MultOS, která slouží jako strana uživatele v revokačním schématu RKVAC. Práce [8] byla zaměřena na vývoj aplikace pro stranu ověřovatele, vydavatele a revokační autority. Aplikace je napsána pro operační systém Raspbian, který slouží jako platforma pro zařízení Raspberry PI. Práce [9] je zaměřena na implementaci schémat KVAC a RKVAC. Pro schéma RKVAC byly vytvořeny dvě aplikace. První je zjednodušenou aplikaci pro platformu MultOS, která zabezpečuje fungování entity uživatele. Druhá aplikace funguje jako entita ověřovatele, vydavatele a revokační autority a je vytvořena pro platformu Raspberry PI.

Webové aplikace, které byly vytvořené v této diplomové práci, využívají obě aplikace vytvořené v rámci práce [9]. Z tohoto důvodu bude součástí této kapitoly analýza vytvořených aplikací pro všechny entity protokolu RKVAC v rámci práce [9].

2.1 Vývojové prostředky

V této části se zaměříme na prostředky, které byly použity pro vývoj aplikací v práci [9].

2.1.1 Čipové karty a MultOS

Stranu uživatele v implementaci reprezentují čipové karty s mikroprocesorem, jako výkonově omezená zařízení. Výhodou těchto karet je kromě možnosti uložení dat, schopnost provádět kryptografické operace. To může být využito na implementování autentizačních technik, jenž poskytují vyšší bezpečnost.

Operační systém, který byl zvolený v analyzovaných implementacích, je MultOS. Jedná se o volně šířený multiaplikační operační systém pro čipové karty, který především umožňuje používat programovací jazyk C. MultOS od verze 4.2. obsahuje nástroje pro základní kryptografické operace, včetně funkcí pro modulární aritmetiku, hashovací funkce (SHA-256, atd.), asymetrickou kryptografii (RSA, ECDH, ECDSA, atd.), symetrickou kryptografii (AES, 3DES, atd.) a kryptografii na bázi eliptických křivek. Díky tomu se jedná o vhodný nástroj pro implementaci schématu atributové autentizace.

2.1.2 Raspberry PI

Primární platformou pro aplikaci entit vydavatele, revokační autority a ověřovatele je Raspberry PI. Jedná se o zařízení o velikost bankovní karty, které obsahuje poměrně výkonný mikropočítač. Primárním účelem těchto zařízení je podpora rozvoje programovacích schopností studentů. V implementacích byly použity modely Raspberry Pi 2 Model B a Raspberry Pi 3 Model B+ [10].

Mikropočítač obsahuje čtyř-jádrový Broadcom procesor s 64-bitovou ARM architekturou. Pro připojení do sítě slouží Gigabit Ethernet port, nebo bezdrátový přijímač WiFi signálu. Zařízení též umožňuje komunikaci přes Bluetooth. Přenos obrazu je možný připojením monitoru přes port HDMI, nebo připojením originální obrazovky přes port DSI. Operační systém je uložen na micro-SD kartě spolu s uživatelskými daty. Na tuto kartu je nutné operační systém nahrát předem z jiného zařízení.

Nejčastěji používaným operačním systémem pro Raspberry Pi je systém Raspberry Pi OS, dříve nesoucí název Rasbian [11]. Jedná se o 32-bitovou open-source distribuci Linux-based systému pro zařízení Raspberry Pi. Vývojově vychází z distribuce Debian. V nejvyšší verzi ponouká jak textové, tak grafické rozhraní. Pro instalaci slouží nástroj Noobs, který zároveň připraví micro-SD kartu, pro fungování se zařízením Raspberry Pi.

Podporovanými systémy pro tuto aplikaci jsou také různé distribuce Linuxu. V této diplomové práci byla aplikace využívána na operačním systému Ubuntu Server LTS.

2.1.3 Komunikace s kartami

Komunikace mezi kartou s mikroprocesorem a terminálem probíhá na bázi server-klient. Model komunikace se řídí standardem ISO 7816, který je založen na podobném principu jako model ISO/OSI. Druhá aplikace, která je v roli klienta, posílá kartě příkazy a ta na ně reaguje odpovědmi. Celá komunikace probíhá přes APDU (Application Protocol Data Unit) zprávy. Zprávy jsou nejčastěji přenášeny pomocí jednoho z dvou následujících transportních protokolů [12]:

- $T = 0$ – poloduplexní bajtově orientovaný protokol, s nízkými paměťovými nároky (300 B), který přenáší pole bajtů s maximální velikostí řídících dat 255 B,
- $T = 1$ – poloduplexní blokově orientovaný protokol, s vyššími paměťovými nároky (1100 B), který přenáší pole bajtů s maximální velikostí řídících dat 65 535 B.

Pro účely této diplomové práce byla komunikace s kartami rozšířená o možnost posílaní APDU zpráv přes TCP protokol pomocí metody TCP Socketing. Tato metoda

byla využita pro komunikaci s kartami, které nejsou připojeny přímo k terminálu s RKVAC aplikací, ale jsou přes čtečku karet připojeny k počítači klienta (viz sekce 3.2.2).

2.2 Entita uživatele

V rámci implementace [9] byla pro entitu uživatele vytvořena aplikace pro platformu čipových karet MultOS ML4. Programovacím jazykem byl zvolený jazyk C a MultOS Assembler. Aplikace vhodně ošetřuje komunikace pomocí APDU zpráv.

Aplikace implementuje důležité funkce pro vydání atributů a jejich pověření na kartu. Zároveň poskytuje kryptografický základ pro revokovatelnost uživatele a komunikaci s revokačí autoritou. Výpočetně nejnáročnější jsou funkce pro komunikaci s entitou ověřovatele. V této implementaci nebyla řešena možnost volby atributů, kterých znalost bude prokázána, ale jenom omezení jejich počtu. Zároveň jsou všechny atributy považovány za veřejné a proto nejsou ověřovateli po ověření posílány.

2.3 Entita vydavatele, ověřovatele a revokační autority

Aplikace pro zbylé tři entity je napsána v jazyce C++ a je určena pro platformu Raspbian GNU/Linux. Obsahuje funkce, které implementují pokročilé kryptografické metody pro výpočet všech dílčích částí protokolu RKVAC (viz sekce 1.3). Zároveň jsou zde definovány funkční nástroje pro posílání a příjem APDU zpráv. Ty zabezpečují veškerou komunikaci s čipovými kartami.

Obsluha aplikace probíhá přes příkazový řádek. Při prvním použití v systému je nutné aplikaci kompilovat a vytvořit spustitelný binární soubor. K tomu je využíván nástroj *cmake*, který je jedním ze systémových požadavků pro chod aplikace.

Po úspěšném vytvoření spustitelné aplikace, je možné ovládat funkce pro jednotlivé entity jejím spuštěním s vhodnými parametry. Při prvním spuštění aplikace vygeneruje potřebné soubory pro jednotlivé entity (páry klíčů, atd. – viz sekce 1.3). Všechny soubory vytvořené během fungování aplikace jsou ukládány do složky `./data`.

V rámci této diplomové práce byla tato aplikace implementována, jako nástroj, primárně určený pro provádění jednotlivých protokolů schématu RKVAC. K této aplikaci bude v práci odkazováno jako k **RKVAC aplikaci**.

3 Webové aplikace

Cílem této diplomové práce bylo vytvořit webové aplikace pro jednotlivé entity protokolu RKVAC. Základním požadavkem bylo, aby aplikace umožňovali konkrétní entitě provádět veškeré funkce, které z její role vycházejí. Pro použitelnost v reálným prostředí bylo nutné aplikace navzájem propojit tak, aby veškerá komunikace probíhala na úrovni webových aplikací a nebylo nutné běžné kroky řešit na úrovni příkazového řádku.

V této kapitole jsou popsány klíčové funkce vytvořených aplikací pro entity vydavatele, revokační autority a ověřovatele (viz sekce 3.3, 3.5, 3.4 a 3.7). Předchází tomu výčet použitých prostředků k vývoji, s uvedenými důvody použití vybraných nástrojů (viz sekce 3.1). V sekci 3.2 je podrobně popsána architektura celého systému založená na modelu klient-server. Funkce a metody sloužící k propojení aplikací do jednotného uceleného systému jsou detailně popsány v sekci 3.6.

3.1 Vývojové prostředky

V této sekci budou popsány prostředky, které byly použity pro vývoj aplikací. Aplikace byla vyvíjená primárně pro 64 bitové distribuce Linuxu. V případe splnění základních požadavků je možné aplikaci spustit na jakémkoliv systému. Mezi základní požadavky patří podpora prostředí Node.js a podpora aplikace RKVAC, která využívá dostupné knihovny primárně určené pro UNIX distribuce. Během vývoje byly aplikace testovány v systémech Ubuntu x86_64 a Ubuntu ARM64.

3.1.1 Node.js

Aplikace jsou vyvíjeny v jazyce JavaScript. Jedná se o jeden z nejpoužívanějších a nejrozšířenějších programovacích jazyků pro vývoj webů a proto je vhodným pro účely vyvíjených aplikací. Z důvodu fungování aplikací v rolích webových serverů (viz sekce 3.2) bylo nutné vytvořit kód jak pro serverovou, tak pro klientskou část. Serverová část, též označována jako **back-end**, je veškerý kód aplikace, který běží na serveru. Klientská část, neboli **front-end**, je kód, který je spouštěn prohlížečem a přímo reaguje na činnost uživatele aplikace. Kvůli tomuto charakteru aplikace bylo použito prostředí Node.js [13].

Jedná se o volně dostupné, multi-platformové prostředí, které poskytuje velké množství nástrojů pro vývoj serverové části aplikace v jazyce JavaScript. Node.js používá nástroj NPM (Node package manager), který poskytuje stovky tisíc balíčků implementujících nástroje pro různé funkce webových aplikací. To umožňuje efektivně vytvářet mnoho různých serverových nástrojů.

Jednou z velkých předností prostředí Node.js je jeho asynchronita. Veškeré procesy jsou tímto prostředím primárně spouštěny paralelně, co v praxi znamená, že jeden proces není blokován druhým. Aplikace je díky tomu vhodná pro větší zátěže, jelikož dokáže vykonávat velké množství stejných úkonů v jednom čase.

Několik klíčových nástrojů pro vývoj webových aplikací však není přímo implementováno v prostředí Node.js. Jedná se například o zpracovávání HTTP (Hypertext Transfer Protocol) a HTTPS (Hypertext Transfer Protocol Secure) požadavků, směrování webů na základě URL (Uniform Resource Locator) řetězců, nebo poskytování statických souborů – například HTML (Hypertext Markup Language) stránek. Místo psaní veškerého kódu je možné použít nějaký z mnoha Node.js frameworků. Na vývoj aplikací byl použitý framework Express [14]. Jedná sa o nejrozšířenější rozhraní pro Node.js. Mezi jeho nejpoužívanější funkce patří široká podpora zpracování HTTPS požadavků, směrování statických souborů a používání šablon pro serverové odpovědi. Express také obsahuje mezivrstvu (middleware), která poskytuje pokročilé nástroje pro zpracování požadavků na server.

3.1.2 Smart Card Browser Extension

Jako bude popsané v sekci 3.2.2, aplikace musí řešit komunikaci mezi webovým prohlížečem a čipovou kartou, připojenou přes čtečku k počítači uživatele. Pro tento účel aplikace implementují nástroj Smart Card Browser Extension, dostupný z [15]. Jedná se o volně dostupný doplněk pro prohlížeč Google Chrome, který umožňuje JavaScriptové aplikaci, běžící v prohlížeči, komunikovat s čipovými kartami připojenými k počítači prostřednictvím APDU zpráv.

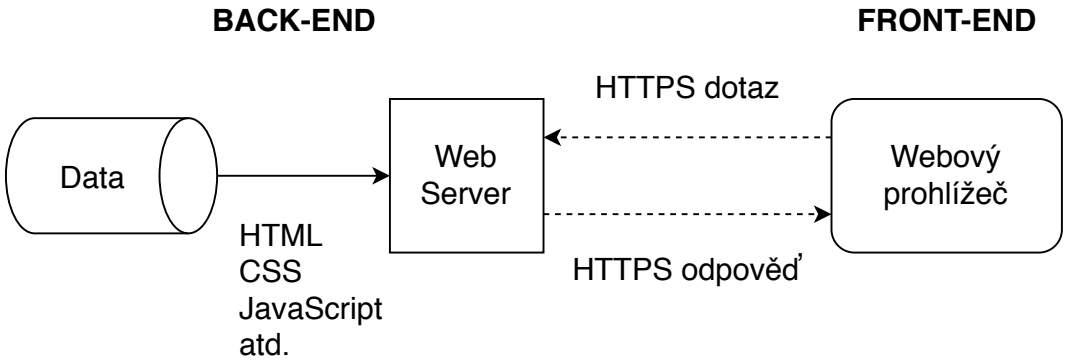
3.1.3 W3.CSS

Pro definici vzhledu webových stránek je používán jazyk CSS. Pro jednodušší implementaci různých stylů byla v aplikacích použita šablona W3.CSS. Jedná se o CSS šablonu, která poskytuje dizajnové styly pro většinu elementů webových stránek. Šablona je veřejně dostupná a je zcela nezávislá od konkrétních JavaScript knihoven. Za její vznikem stojí firma W3 Schools, která poskytuje výukovou platformu pro programátory [16].

3.2 Architektura systému

V této sekci bude blíže popsána architektura vytvořeného systému. Veškeré principy uvedené v této sekci jsou stejné pro všechny tři aplikace, které byly v rámci diplomové práce vytvořeny.

Webové aplikace fungují jako webové servery. Komunikace v rámci jedné aplikace funguje na bázi klient-server. V roli klienta v tomto procesu vystupuje webový prohlížeč, přes který se uživatel připojuje k rozhraní aplikace. Uživatel přes webový prohlížeč zadává aplikaci úlohy, které jsou následně posílány na server ve formě HTTPS požadavků. Server pracuje s lokálními soubory, spouští implementovanou RKVAC aplikaci a výstupy posílá jako HTTPS odpovědi zpátky webovému prohlížeči. Fungování aplikace jako webového serveru je znázorněno na Obr. 3.1 [17]. Server používá zabezpečenou verzi protokolu HTTP. Celá komunikace mezi klien-



Obr. 3.1: Aplikace jako webový server

tem a serverem je díky tomu šifrována, což přispívá k vyšší bezpečnosti aplikace. HTTPS servery, které jsou vytvářeny jednotlivými aplikacemi, používají porty zobrazené v Tab. 3.1.

Aplikace	**Port**
Vydavatel	10443
Revokační autorita	9443
Ověřovatel	8443

Tab. 3.1: Přehled portů HTTPS serverů

Základní metody protokolu HTTPS používané na komunikaci mezi webovým prohlížečem a serverem jsou:

- `GET` – požadavek na zaslání dat ze serveru. Jedná se o nejpoužívanější metodu protokolu HTTPS. Parametry požadavku jsou na server posílány přímo v URL řetězci.
- `POST` – zpráva obsahující uživatelské data určené pro server. Data jsou vetšinou získány z formulářů dostupných ve webovém rozhraní a jsou posílány v těle požadavku `POST`. Tato metoda je proto výpočetně náročnější.

JavaScript kód každé aplikace je rozdělen na dvě části:

- **Back-End** – Node.js kód, který je spouštěn na serveru přímo operačním systémem,
- **Front-End** – skripty psané v čistém JavaScriptu, které jsou spolu s HTML stránkami posílané klientovi a jsou spouštěny v prohlížeči.

3.2.1 Back-end

Back-endem aplikace se rozumí část, která běží přímo na serveru. Nacházejí se zde veškeré soubory aplikace, které jsou členěny do vhodné struktury (viz příloha E). Základním souborem, který je spouštěn, je `./bin/www`. V tomto souboru je vytvořen HTTPS server, který naslouchá na určeném portu. Základní nastavení aplikace je definováno v souboru `./app.js`, nicméně o veškeré reakce na požadavky klienta se stará router [1] , kterého zdrojový kód se nachází v souboru `./routes/index.js`. Zde je umístěna většina back-end kódu, vytvořena v této práci.

Aplikace přímo pracuje s aplikací RKVAC, která byla vytvořena v práci [9]. Zdrojový kód aplikace je nutné podle dokumentace kompilovat, čímž vznikne spustitelná RKVAC aplikace. Tuto aplikaci je nutné nakopírovat do složky se zdrojovým kódem webové aplikace.

Webová aplikace používá pro práci s aplikací RKVAC modul `Child process`. Jedná se o nativní modul prostředí Node.js, který je schopen spouštět procesy v Shellu. V momentě kdy server přijme požadavek ze strany klienta, který zahrnuje zasáhnutí do RKVAC systému, je spuštěn nový proces, v kterém je RKVAC aplikace zavolána s příslušnými parametry. Po ukončení je RKVAC proces webovou aplikaci vyhodnocen a klient je informován o výsledku. Příklad použití RKVAC aplikace pro personalizaci karty je možné vidět ve výpisu 3.1.

[1] Router je modul prostředí Express.js, který definuje, jak je naloženo s požadavky klienta.

Výpis 3.1: RKVAC child process – personalizace karty

```
router.post('/post-new-user', (req, res) => {
let command = "./rkvac-protocol-multos-1.0.0 -p -n " +
req.body.name + " -s " + req.body.surname;
exec(command, (error, stdout, stderr) => {
    if (error) {
        res.json({success: false});
        //informace o chybě poslána na klienta
        return;
    }
    if (stderr) {
        res.json({success: false});
        //informace o chybě poslána na klienta
        return;
    }
    res.json({success: true});
    //informace o úspěchu poslána na klienta
});
```

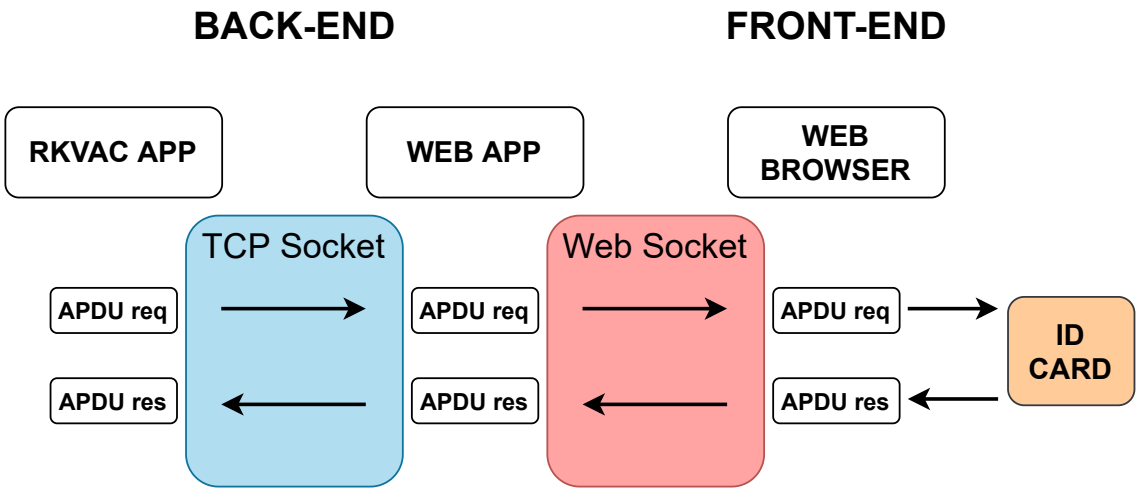
3.2.2 Komunikace RKVAC-karta

Aplikace RKVAC je primárně vytvořená pro platformu Raspberry PI a předpokládá přímé připojení čtečky s ID kartou k terminálu, na kterém aplikace běží. Webové aplikace vytvořené v této práci pracují s uživatelem vzdáleně a proto bylo nutné upravit komunikaci tak, aby byla RKVAC aplikace na back-endu schopna komunikovat s ID kartou připojenou k počítači uživatele.

Pro přenos APDU zpráv mezi RKVAC aplikací a ID kartou byl použit koncept s názvem **TCP Socketing**. Jedná se o způsob komunikace mezi dvěma stranami, na základě posílání přímých zpráv pomocí socketů protokolu TCP. Princip spočívá v tom, že jedna strana vytvoří socket, který naslouchá na určené IP adrese a portu, a vystupuje jako server. Druhá strana se pomocí svého socketu na server připojí, čímž vznikne přímé TCP spojení. Přes toto spojení jsou následně obě strany schopny vyměňovat si zprávy, ve formátu `buffer`.

V případě, že RKVAC aplikace chce navázat spojení s ID kartou, vytvoří TCP socket server na adrese localhost na daném portu (pro každou webovou aplikaci jiný, viz Tab. 3.2) a čeká na připojení druhé strany. Na toto spojení se připojí webová aplikace běžící na back-endu, která v tomto scénáři vystupuje jako prostředník mezi RKVAC aplikací a ID kartou. Obě aplikace běží na stejném serveru a dokáží proto komunikovat na localhostu. V momentě kdy se webová aplikace na TCP socket RKVAC aplikace připojí, je jí poslána APDU zpráva pro ID kartu. Webová aplikace

má na stejným principu připravený druhý socket server, na který se připájí klient – webový prohlížeč. Pomocí tohoto spojení přepošle APDU zprávu od RKVAC aplikace na klienta, který ji pomocí doplňku Smart Card Extension (viz sekce 3.1.2) pošle na ID kartu připojenou k počítači uživatele. Jakmile klient obdrží od karty odpověď, přepošle ji zpátky na back-end webové aplikace, která ji pomocí prvního socket spojení přepošle na RKVAC aplikaci. Stejným způsobem je naloženo s veškerou APDU komunikací mezi RKVAC aplikaci a ID kartou. Poslání jednoho APDU požadavku a přijetí odpovědi je znázorněno na Obr. 3.2.



Obr. 3.2: APDU komunikace

Spojení mezi back-endem a front-endem webové aplikace využívá místo čistého TCP spojení protokol **Websocket** [18], který slouží k vytváření socket spojení přes web. Websocket server běží přímo na vytvořeném HTTPS serveru a celý přenos dat je proto šifrován.

Port pro TCP spojení mezi RKVAC aplikaci a back-endem webové aplikace je jiný pro každou entitu protokolu RKVAC. Důvod na oddělení jednotlivých spojení je předcházení kolizím pokud více aplikací běží na jednom serveru. Přehled použitých portů ve vytvořených aplikacích je možné vidět v Tab. 3.2.

Aplikace	**Port**
Vydavatel	5002
Revokační autorita	5001
Ověřovatel	5000

Tab. 3.2: Přehled portů TCP socketů

3.3 Aplikace vydavatele

V této sekci bude popsána první vytvořená aplikace v této diplomové práci, a to aplikace pro entitu vydavatele. Jak je patrné z části 1.3, aplikace musí primárně poskytovat funkce pro vydání atributů na čipovou kartu. Zároveň je zde umístěn modul pro personalizaci nové karty, kdy je na kartu vydáno identifikační číslo.

3.3.1 Personalizace karty

Prvním krokem integrace čipové karty do RKVAC systému je personalizace karty, kterou je spustit přes stránku „Users“. Na této stránce se nachází tabulka „User list“ a panel „New User“. Pro personalizaci karty je nutné zvolit příslušnou čtečku ze seznamu na kartě „New User“, zadat jméno a příjmení vlastníka karty a kliknout „Personalize“. Po kliknutí je odeslán HTTP POST požadavek na server, který v těle obsahuje zadané hodnoty v tvaru JSON (JavaScript Object Notation). Server extrahuje ze zprávy data a se zadanými parametry spustí RKVAC aplikaci, která kartu personalizuje.

Seznam uživatelů v systému je uložen v souboru `./data/Issuer/user_list.csv`. Aktuální stav tohoto souboru je při načtení stránky zapsán do tabulky „User list“. Seznam je možné aktualizovat také pomocí tlačítka, přičemž čas poslední aktualizace je zapsán nad tabulkou.

3.3.2 Vydání atributů

Hlavní úlohou vydavatele je vydávat na ID karty atributy. Tato funkce je v aplikaci dostupná na stránce „Credentials“. Atributy jsou na kartu vydávány ve formě pověření, které obsahuji názvy atributů a jejich hodnoty. Při ověřování se ověřuje atribut podle pozice, na které se má v rámci pověření nacházet. Ve výsledku proto záleží na hodnotě atributu a jeho pozici, a ne na jeho názvu.

Pro zachování univerzálního formátu, aplikace podporuje 3 druhy pověření, které mají definovány počet atributů a jejich názvy. Jedná se o:

- **EID** s atributy:
  - Name and Surname,
  - Birthdate,
  - Nationality,
  - Permanent residence,
  - Sex.
- **Ticket** s atributy:
  - Name and Surname,
  - Card number,

  - Type of ticket.
- **Employee's card** s atributy:
  - Name and Surname,
  - Employee ID,
  - Employer,
  - Employee position.

Vydavatel má také možnost vytvořit vlastní pověření s 1-9 atributy, přičemž těmto atributů jsou automaticky přiděleny jména „Attribute1-9".

Vydavatel má dva způsoby vydání pověření na kartu. Prvním je vytvoření nového pověření přes panel „New credentials". Zde je nutné zvolit jeden z připravených formátů pověření, nebo si vytvořit vlastní. Po zadaní názvu pověření a jednotlivých atributů, je požadavek odeslán na server, kde je spuštěn protokol `IssueI` (viz sekce 1.3.2). Po proběhnutí tohoto protokolu je pověření vydáno na zvolenou kartu. Pověření je na straně serveru uloženo do souboru se zadaným názvem a s příponou `.att`.

Druhým způsobem vydání pověření je vydání jednoho z existujících pověření ze seznamu „List of credentials". Tento list je pravidelně aktualizován a ukazuje dostupné soubory s příponou `.att` ve složce `./data/Issuer/`. Po zvolení pověření a příslušné čtečky je obdobným způsobem, jako v prvním případě, pověření vydáno na kartu. Pověření v seznamu „List of credentials" je možné vymazat a také si zobrazit jejich detail.

Prvním krokem protokolu `IssueI` je ověření platnosti revokačního handleru vydaného na kartě. Toto se provede pomocí veřejného klíče revokační autority, která handler vydala (viz 3.5.1). Pro úspěšné vydáni atributů je proto nutné přes panel „Tools" na domovské stránce aplikace, nahrát veřejný klíč revokační autority do systému vydavatele. Jedná se o soubor `ra_pk.dat` a je možné ho získat přes aplikaci revokační autority (viz 3.5.1).

Při prvním vydávání atributů je spuštěn protokol `SetupI` (viz 1.3.1), během kterého je vygenerován soukromý klíč vydavatele. Tento klíč je ke stažení pro další použití na domovské stránce v panelu „Tools". V případě migrace entity vydavatele z jiného systému, je možné jeho soukromý klíč nahrát do systému předem. V tomto případě protokol `SetupI` není spuštěn a použije se nahraný klíč.

3.4 Aplikace ověřovatele

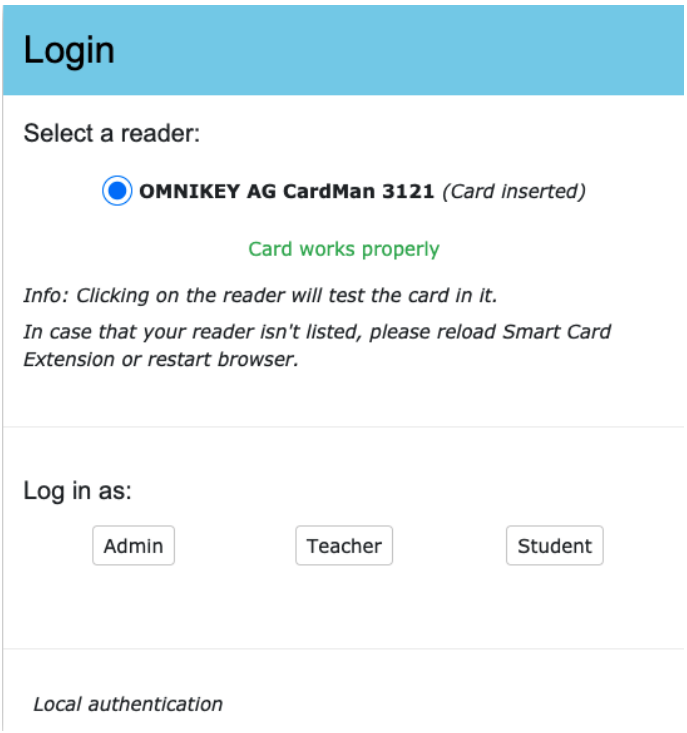
Druhou aplikací, která byla v této diplomové práci vytvořena, je aplikace pro entitu ověřovatele. Hlavní náplní bylo vytvořit autentizační modul založený na systému RKVAC, který bude chránit webovou službu. Jako příklad webové služby bylo vytvořeno rozhraní pro přístup k databázi vysokoškolských předmětů, vyučujících

a ústavů. Samotná databáze je vytvořena v databázovém programu MongoDB [19] a je uložena na jeho oficiálních serverech. Aplikace ověřovatele se k databázi připojuje a poskytuje funkce pro její ovládáni, jako jsou prohlížení, vytváření, úprava a mazání objektů. Objekty jsou vhodně propojeny, aby bylo možné přidělovat jednotlivé předměty vyučujícím a určovat ústavy, na kterých jsou vyučovány.

Tato kapitola bude dále pojednávat pouze o autentizačním modulu, který řídí přístup k aplikaci. Kromě autentizačního modulu založeného na RKVAC je možné se vůči aplikaci autentizovat pomocí jména a hesla viz 3.7.1. Tato metoda je určená pro administrátory aplikace a slouží na přihlášení za účelem přípravy systému RKVAC.

3.4.1 Autentizační modul

Přístup do aplikace je chráněn autentizačním modulem, který je založen na schématu RKVAC. Modul implementuje entitu ověřovatele a povoluje přístup na základě poskytnutých důkazu o držení určených atributů. Při připojení k aplikaci přes webový prohlížeč se uživateli zobrazí přihlašovací stránka (viz Obr. 3.3). Po zvolení čtečky



Obr. 3.3: Přihlašovací stránka

se vsunutou ID kartou a kliknutí na jednu z nabízených přístupových rolí, je na server odeslán HTTPS POST požadavek. Na straně serveru je z požadavku extrahována informace o vyžádané přístupové roli a je spuštěn ověřovací protokol (viz sekce 1.3.3). Ověřovatel má v systému přichystané pověření, které definuje atributy

a jejich pozice v rámci pověření, které jsou během ověřování od uživatele vyžádány. Pokud se uživateli podaří prokázat držení vyžádaných atributů na daných pozicích v rámci jeho pověření, je úspěšně autentizován a je mu udělen přístup do aplikace. V opačném případě je přístup zamítnut a uživatel je o tom patřičně informován. V případě kdy dojde k chybě způsobené na straně RKVAC, je na přihlašovací stránce zobrazena příslušná hláška.

Před prvním použitím je nutné autentizační modul nastavit. Veškeré nastavení se provádí přes stránku „Setup“ v aplikaci. Prvním krokem je nahrání soukromého klíče vydavatele a veřejného klíče spolu s veřejnými parametry revokační autority. Jedná se o soubory `ie_sk.dat`, `ra_pk.dat` a `ra_public_parameters.dat`, a je možné je stáhnout z aplikací příslušných entity.

Druhým krokem je příprava přístupových pověření (viz 3.4.3). Posledním potřebným nastavením je přechod na první epochu a připojení aplikace k revokační autoritě (3.4.4).

Po nastavení všech části je autentizační modul připraven pro použití.

3.4.2 Autorizační role

Kromě autentizace je v aplikaci implementována autorizace s třemi autorizačními rolemi:

- Admin,
- Teacher,
- Student.

Autorizační role definuje oprávnění, které uživatel v aplikaci má. Pro uvedení příkladu jsou jednotlivým rolím přiděleny tyto oprávnění pro práci s objekty databáze:

- Admin:
  - předměty – čtení, zápis,
  - vyučující – čtení, zápis,
  - ústavy – čtení, zápis.
- Teacher:
  - předměty – čtení, zápis,
  - vyučující – čtení, zápis,
  - ústavy – čtení.
- Student:
  - předměty – čtení,
  - vyučující – čtení,
  - ústavy – čtení.

Pro přístup k stránkám „Setup“, „Access Logs“ a „Change password“ má oprávnění jenom administrátor.

3.4.3 Přístupová pověření

Pro každou roli je na serveru připraveno přístupové pověření, které definuje vyžadované atributy a jejich pozice. Nastavení těchto pověření se provádí na stránce „Setup“ v paneli „Access credentials“. Tabulka v tomto panelu ukazuje aktuální stav pověření pro jednotlivé role. V případě, že je pověření vytvořené, v tabulce je označeno jako **ready** a je možné si zobrazit jeho podrobnosti, nebo ho vymazat. V opačném případě je označeno **not ready**.

Vytvoření nového pověření je prováděno v následujících krocích:

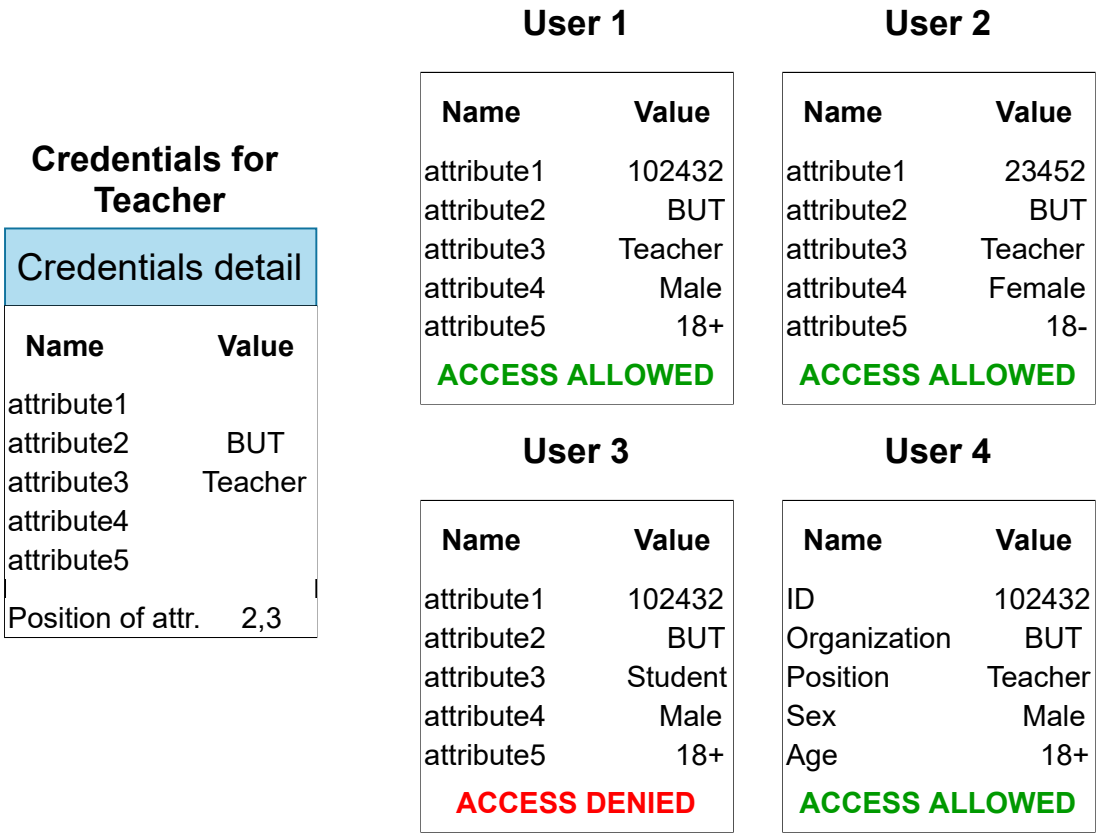
1. Zvolení autorizační role – jedná se o jednu z třech rolí admin, teacher, student.
2. Zvolení počtu atributů v pověření – počet atributů musí být stejný, jako je počet atributů v pověřeních, které jsou v rámci systému vydávané na karty uživatelů.
3. Zadání jednotlivých atributů – na příslušné pozice je nutné zadat atributy, jež budou od uživatele během autentizace vyžádány. Atributy v pověření, které nebudou vyžadovány, je možné nechat prázdné.
4. Určení pozic vyžadovaných atributů – pozice atributů v rámci pověření, které budou během ověřování vyžadovány, se zapíše ve formě čísel (pozic) oddělených čárkou. Jestli je pole nevyplněno, vyžaduje se automaticky atribut na pozici 1.

Po vyplnění potřebných údajů a kliknutí na „Vytvořit“, jsou údaje odeslány na server, kde je vytvořen nový soubor s názvem `DB<role>.att`. Toto pověření se následně použije při ověřovaní atributů během autentizace.

Příklad přístupového pověření je zobrazen na Obr. 3.4. Pro lepší pochopení procesu ověřování jsou zde uvedené příklady konkrétních pověření vydaných uživatelům a výsledek, který můžeme během autentizace očekávat. Jak je z Obr. 3.4 patrné, **User 1** je v pořádku autentizován vůči entitě ověřovatele, jelikož je schopen prokázat držení atributů „BUT“ a „Teacher“ na pozicích 2 a 3. Příklad **User 2** slouží na poukázání na skutečnost, že atributy na jiných pozicích v rámci pověření nehrají v ověřovacím procesu roli a neovlivni výsledek autentizace. Na příkladu **User 3** můžeme vidět, že pro úspěšné ověření musí uživatel disponovat oběma vyžadovanými atributy. Jelikož táto podmínka není splněná, tento uživatel nebude autentizován. V poslední řade příklad **User 4** demonstruje, že při ověřování nezáleží na názvu atributu, ale jenom na jeho hodnotě a pozici. Názvy atributů v rámci pověření mají pouze formální účel.

3.4.4 Epochy

Entita ověřovatele používá pro fungování princip epoch. Ten spočívá v pravidelném přechodu na novou epochu, kterou aktivuje revokační autorita. Epocha je označená



Obr. 3.4: Příklad autentizace vůči přístupovému pověření

8-místním číslem, které je složeno z aktuálního data a z pořadového čísla epochy.

Veškeré funkce pro spravování epochy jsou dostupné v panelu „Epoch settings“ na stránce „Tools“ ve webové aplikaci. Je zde uvedené číslo aktuální epochy s možností okamžitého přechodu na novou. Pro správné fungování RKVAC systému je nutné, aby byla epocha aktivována revokační autoritou. Za tímto účelem je zde možnost uložení IP adresy serveru, na kterém běží aplikace revokační autority. Po uložení adresy je možnost přechodu na novou epochu aktivována. Komunikace mezi aplikacemi ověřovatele a revokační autority je popsána v sekci 3.6.2.

V rámci jedné epochy je omezen počet ověřovacích cyklů, které můžou být ověřovatelem uskutečněny. Je proto doporučené pravidelně měnit epochu a tím udržovat systém aktuální a připraven pro použití. Pro tento účel je v panelu „Epoch settings“ dostupný časovač přechodu na novou epochu. Správce aplikace tak může naplánovat pravidelný automatizovaný proces přechodu na novou epochu. Časovač se zadává ve formátu **crontab**, což je softwarový démon, který v operačním systému spouští nějaký automatizovaný proces [20]. Záznam v tomto formátu se skládá z pěti čísel oddělených čárkou. Každé číslo definuje jinou časovou jednotku – „`<minuta> <hodina> <den(v měsíci)> <měsíc> <den(v týdnu)>` “, přičemž znak `*` definuje hodnotu

„každý“. Pro příklad je uveden záznam `0 0 * * *`, který spouští úlohu vždy v čase 0:00 každý den, každý měsíc.

3.4.5 Přístupové logy

Na stránce „Access logs“ se nachází výpis ze souboru `ve_requests.log`, který v sobě obsahuje záznamy ověřovacích cyklů. Pro každý záznam jsou zde dostupné položky:

- datum,
- čas,
- číslo epochy,
- pseudonym uživatele C,
- výsledek ověření – povolení nebo zmítnutí přístupu.

Logy jsou seřazeny od nejnovějších a jejich počet je omezen na 50 záznamů. V případě potřeby je možné dohledat veškeré záznamy přímo na serveru v souboru `ve_requests.log`, který se nachází v složce `./data/Verifier/`.

3.5 Aplikace revokační autority

Třetí aplikací, která byla v rámci této diplomové práce vytvořena, je aplikace pro entitu revokační autority systému RKVAC. Možnost revokace uživatele ze systému je jednou z největších předností systému RKVAC oproti jiným systémům atributové autentizace. Vytvoření aplikace pro tuto entitu bylo proto logickým krokem, bez kterého by systém aplikací nebyl kompletní a neposkytoval by používání všech dostupných funkcí.

V této části budou popsány funkce, které aplikace revokační autority poskytuje a jejich zařazení do systému RKVAC.

3.5.1 Revokační handlery

Revokační autorita je schopna uživatele revokovat na základě revokačního handleru, který mu vydala na kartu. Po personalizaci karty (viz 3.3.1) je nutné kartu připojit k aplikaci revokační autority a tento handler na kartu vydat. To se provede pomocí prvního panelu v aplikaci „Revocation handler“. Na tomto panelu je dostupný výpis připojených čteček karet s možností otestování karty v čtečce správou APDU_SELECT. Po zvolení příslušné čtečky a kliknutí na „Assign“, je zahájen protokol `IssueRA` (viz sekce 1.3.2) a revokační handler je na kartu vydán.

Vydání revokačního handleru je podmínkou pro vydání atributů na kartu vydavatelem. Handler je podepsán soukromých klíčem revokační autority a vydavatel je

proto pomocí její veřejného klíče schopen tento podpis ověřit. V případě, že handler při vydávání atributů na kartě chybí, nebo není možné ověřit podpis revokační autority, atributy nejsou na kartu vydány.

Při prvním vydávání revokačního handleru je spuštěn protokol `SetupRA`, během kterého jsou vygenerovány následující soubory:

- soukromý klíč,
- veřejný klíč,
- soukromé parametry,
- veřejné parametry.

Všechny toto soubory je možné následně stáhnout přes panel „Revocation authority's files“ pro další použití. Všechny tyto soubory je možné předem do systému nahrát. V takovým případě nebude konkrétní nahraný soubor vygenerován znovu.

3.5.2 Revokace uživatelů

Primární funkcí revokační autority je schopnost revokovat uživatelé ze systému. Tyto uživatelé následně nebudou schopny se ověřit vůči ověřovatelům, jelikož se budou nacházet na blacklistu a jejich žádosti budou automaticky zamítnuty. Pro revokaci slouží panel „Revocation of a user“ a existují dva způsoby revokace uživatele.

- Revokace na základě identifikátoru – do pole „User's identifier“ se zadá ID uživatele, které mu bylo přiděleno během personalizace.
- Revokace na základě pseudonymu C a čísla epochy – tyto údaje je možné zjistit z přístupových logů ověřovatele (viz 3.4.4).

Po zadaní parametrů pro jednu ze dvou možností je nutné zadat IP adresu ověřovatele. Je možné zadat i více adres oddělených čárkou. Po zadaní adresy a kliknutí na „Revokovat“ je uživatel revokován ze systému a aktualizovaný blacklist je poslán na všechny zadané adresy ověřovatelů.

Zadáním více adres je možné spravovat pomocí revokační autority větší systém, ve kterém figuruje mnoho entity ověřovatelů. Aplikace v této situaci prvně revokuje uživatele a následně naváže potřebný počet spojení na ověřovatele, pomocí kterých jim pošle aktualizovaný blacklist. Počet paralelních spojení je omezen řádově na tisícky spojení. Toto mimořádné zvládání zátěže je dáno vlastnostmi prostředí Node.js, které je pro takový typ úloh ideální (viz 3.1.1).

3.5.3 Seznam ověřovatelů

Při změně epochy na straně ověřovatele je číslo epochy posláno na aplikaci revokační autority, která tuto epochu musí aktivovat (viz 3.4.4). Fungování spojení mezi aplikací ověřovatele a revokační autority je blíže popsáno v části 3.6.2.

Pro úspěšnou změnu epochy ze strany ověřovatele je nutné, aby byla IP adresa ověřovatele na whitelistu revokační autority. Úprava tohoto whitelistu je možná přes stránku „Verifiers“, dostupnou přes vrchní lištu aplikace.

3.6 Propojení aplikací

Všechny aplikace vytvořené v této práci fungují jako ucelený systém. Každá entita má v schématu RKVAC své místo a je potřebná pro bezchybné fungování. V této části je popsáno propojení aplikací, které bylo v malé míře načrtnuto v předchozích částech.

3.6.1 Management klíčů

První části propojení aplikací je zabezpečení přístupu jednotlivých entit k potřebným souborům ostatních entit. V každé aplikaci se nachází panel, pomocí kterého je možné tyto soubory spravovat. V následujícím seznamu jsou uvedeny všechny soubory, se kterými je nutno v rámci systému pracovat.

- Soukromý klíč vydavatele – jedná se o soubor `ie_sk.dat`, který je generován během protokolu `SetupI` (viz. 1.3.1). Tento protokol je spouštěn během prvního vydávání atributů vydavatelem a je uložen v jeho systému ve složce `./data/Issuer/`. Tento klíč je používán také entitou ověřovatele, jelikož tyto entity sdílí soukromý klíč. Po vygenerování je nutné soubor stáhnout přes panel „Tools“ v aplikaci vydavatele, a přes panel „RKVAC keys“ v aplikaci ověřovatele jej importovat do systému ověřovatele.
- Veřejný klíč revokační autority – jedná se o soubor `ra_pk.dat`, který je generován během protokolu `SetupRA` (viz. 1.3.1). Tento protokol je spouštěn během prvního vydávání revokačního handleru revokační autoritou a je uložen v její systému ve složce `./data/RA/`. Tento klíč je používán také ověřovatelem, který jej používá pro ověření důkazů znalosti pseudonymů, které mu během protokolů `Show` a `Verify` posílají uživatelé. Po vygenerování je nutné soubor stáhnout přes panel „Revocation authority's files“ v aplikaci revokační autority, a přes panel „RKVAC keys“ v aplikaci ověřovatele jej importovat do systému ověřovatele. Kromě ověřovatele používá tento klíč také vydavatel, který pomocí něj ověřuje platnost podpisu revokačních handlerů uživatelů, při vydávání atributů. Je proto nutné jej přes panel „Tools“ v aplikaci vydavatele nahrát i do jeho systému.
- Veřejné parametry revokační autority – jedná se o soubor `ra_public_parameters.dat`, který je generován během protokolu `SetupRA`

(viz. 1.3.1). Tento soubor používá také ověřovatel, pro stejné účely jako veřejný klíč revokační autority. Je proto nutné ho stejným způsobem do systému ověřovatele importovat.

Pokud tyto soubory budou importovány do jiných aplikací, které implementují entity systému RKVAC, tak budou aplikace navzájem kompatibilní.

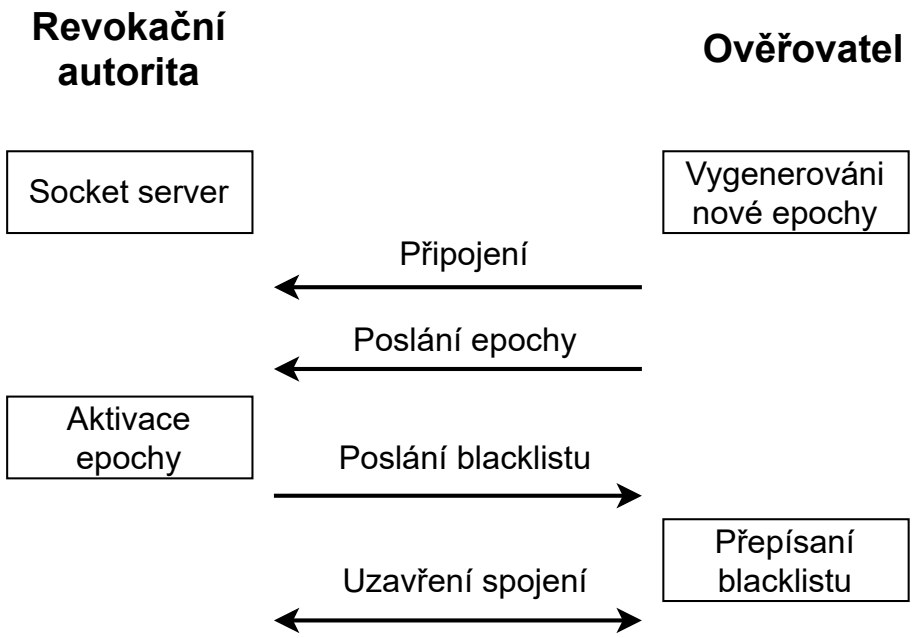
3.6.2 Komunikace RA-Ověřovatelé

Během procesu změny epochy a revokaci uživatele dochází k přímé komunikace mezi aplikaci revokační autority a ověřovatele. V procesu této komunikace si musí aplikace vyměňovat větší počet dat, s kterými se následně na obou stranách pracuje. Pro tento účel implementují aplikace TCP socketing, který byl popsán v části 3.2.2. Sled událostí, které jsou vykonány během jednotlivých procesů, jsou popsány v této sekci.

Prvním krokem při změně epochy, kterou iniciuje ověřovatel, je vygenerování nové epochy a zahájení připojení k aplikaci revokační autority. Tato aplikace vytváří TCP socket server na portu 5004, na který se ověřovatel připojí a pomocí tohoto spojení pošle autoritě číslo epochy. Spojení jsou akceptovány jenom z IP adres, které jsou uvedeny ve whitelistu (viz 3.5.3). Po přijatí čísla epochy, je tato epocha aktivována revokační autoritou. Během aktivace je vygenerován nový blacklist pro konkrétní epochu, do kterého jsou z globálního blacklistu importovány veškeré záznamy a je vytvořen soubor pro ověřovatele. Tento soubor je po řádcích poslán pomocí otevřeného TCP spojení zpátky na ověřovatele, přičemž po skončení přenosu je spojení uzavřeno. Na straně ověřovatele je po skončení spojení aktivována funkce, při které dojde k přepsání aktuálního blacklistu na základě dat přijatých od revokační autority. Celý proces změny epochy a propojení těchto dvou aplikací během procesu je vidět na Obr. 3.5.

Revokace uživatele funguje na stejným principu jako změna epochy, s rozdílem že TCP server vytváří ověřovatel na portu 5003 a revokační autorita se po revokaci uživatele na tento server připojí. Pomocí spojení je nový blacklist poslán na ověřovatele, který jím přepíše svůj aktuální. V tomto případě ověřovatel přijme spojení jenom z adresy, kterou má uloženou jako adresu revokační autority (viz 3.4.4).

Během vývoje byly tyto principy implementovány na stranu aplikace ověřovatele jiného studenta, která následně byla schopna komunikovat s aplikací revokační autority vytvořené v této práci. Bylo tak prokázáno, že aplikace revokační autority může sloužit jako centrální prvek v systému s větším počtem ověřovatelů.



Obr. 3.5: Proces změny epochy

3.7 Globální funkce

V této sekci jsou popsány funkce, které jsou implementované ve všech aplikacích a nejsou specifické jenom pro jednu z entit protokolu RKVAC.

3.7.1 Lokální autentizace

Každá z aplikací je chráněná autentizací jménem a heslem. Autentizace probíhá klasickým způsobem, kdy uživatel zadá do formuláře přihlašovací jméno a heslo a vyžádá si přístup do aplikace. Na serveru je zadané jméno a heslo porovnáno se vzorem, a pokud jsou zadané údaje správné, uživatel je autentizován. Hesla jsou zahashována funkcí SHA-256 (Secure Hash Algorithm), aby byly chráněny proti případnému úniku. Heslo je v této podobě uloženo v souboru `./passwd`. Pro vytvoření autentizační strategie je používán Node.js modul Passport.js [21].

Každá aplikace má jeden uživatelský účet, který má plné oprávnění k používání aplikace. Heslo je možné pro tento účet změnit pomocí stránky „Change password“. Ověřování autentizovaného uživatele je nastaveno pro každý požadavek, který je na server posílán. Aplikace jsou tak chráněny před útoky, kdy se útočník snaží odesílat různé `GET` a `POST` pakety a tím obejít uživatelské rozhraní.

V případě, že více aplikací běží na stejným serveru, tak přihlášení do jedné z nich automaticky uživatele odhlásí z ostatních. Toto je způsobeno ukládáním cookies pro aktuální autentizační cyklus v prohlížeči Google Chrome. Částečně je možné tuto

situaci eliminovat připojením se k jiným aplikacím přes inkognito mód, avšak pro všechny aplikace v rámci tohoto módu je situace stejná.

Z aplikace je možné se odhlásit pomocí příslušného tlačítka v horní liště.

3.7.2 Logování

Výstup ze všech důležitých funkcí je ukládán do souboru `./main.log`. Před každý záznam je vložena časová značka, aby bylo možné dohledat přesné datum a čas, kdy byla funkce vykonána. Časová značka je vkládána ve formátu RRRR/MM/DD HH:MM:SS.

Soubor `./main.log` je možné přes grafické rozhraní aplikací stáhnout, s použitím ikony stažení v horní liště.

3.7.3 Reset aplikace

Každá z aplikací může být resetována do původního stavu. Funkce pro reset se nachází na hlavních stránkách v panelu „Application reset“ a po kliknutí je vyžadováno od uživatele potvrzení o úmyslu vymazání. Během resetu je vymazána složka `./data/`, která v sebe obsahuje veškeré soubory systému RKVAC. Soubory mimo této složky nejsou změněny a nedojde proto k změně hesla na původní hodnotu, ani k přemazání logů.

Závěr

Zadáním této diplomové práce bylo vytvořit webové aplikace, které budou implementovat veškeré funkce pro jednotlivé entity schématu RKVAC. V první části práce jsme se věnovali teoretickému popisu tohoto schématu a rozboru jednotlivých protokolů, které jsou v rámci něho implementovány. V druhé části jsme na základě analýzy současných prostředků zvolili základ, na kterém jsme během vývoje stavěli. Třetí část práce se zaměřuje na popis vlastního přínosu, kterého bylo dosáhnuto.

V práci byly vytvořeny tři aplikace, pro vydavatele, ověřovatele a revokační autoritu. Tyto entity jsou pomocí aplikací schopny vykonávat svou roli v plné míře a ovládat všechny funkce, které jím z jejich podstaty vyplývají. Aplikace ověřovatele poskytuje autentizační modul, který je univerzálně použitelný pro zabezpečení většiny webových služeb, které jsou kompatibilní s prostředím Node.js. Aplikace vydavatele je kompatibilní s jakýmikoliv implementacemi schématu RKVAC a je proto možné ji využívat jako centrální entitu vydavatele v systémech. Aplikace revokační autority implementuje funkce pro správu systémů s velkým počtem ověřovatelů a pokud se do jejich aplikací implementují funkce popsáné v části 3.6.2, je možné ji využívat jako centrální prvek v reálných systémech.

V přílohách práce je možné nalézt instalační a uživatelský manuál. Je zde přítomen také důkaz znalosti funkčnosti aplikací. Pomocí vytvořeného demonstrátoru si kdokoliv může veškeré funkce vyzkoušet a seznámit se s fungováním aplikací.

Aplikace poskytují vhodné uživatelské rozhraní a jsou optimalizovány. Veškerý provoz mezi serverem a klientem je šifrován pomocí protokolu HTTPS. Během vývoje bylo myšleno na bezpečnost aplikací a jsou proto konstruovány s důrazem na ochranu soukromí a integrity dat.

Literatura

[1] EVROPSKÁ UNIE, 27. dubna 2016n. *Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES*. In: . Brusel: Úřední věstník Evropské unie, ročník 2016, číslo 679. Dostupné také z: `https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=celex%3A32016R0679`

[2] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. *Handbook of applied cryptography*. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[3] DAMGARD, Ivan. *On Σ-protocols* [online]. V.2. CPT, 2010 [cit. 2020-10-16]. Dostupné z `https://www.cs.au.dk/~ivan/Sigma.pdf`

[4] KOGAN, Dima. *Lecture 5: Proofs of Knowledge, Schnorr's protocol, NIZK* [online]. Spring 2019, , 2-3 [cit. 2020-10-16]. Dostupné z: `https://crypto.stanford.edu/cs355/19sp/lec5.pdf`

[5] CAMENISCH, Jan, Manu DRIJVERS and Jan HAJNY. Scalable Revocation Scheme for Anonymous Credentials Based on n-times Unlinkable Proofs. In: *WPES '16 Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society* [online]. NY, USA: ACM New York, 2016, s. 123-133 [cit. 2020-10-16]. ISBN: 978-1-4503-4569-9. Dostupné z: `https://dl.acm.org/doi/10.1145/2994620.2994625`

[6] CAMENISCH, Jan, Manu DRIJVERS, Petr DZURENDA and Jan HAJNY. Fast Keyed-Verification Anonymous Credentials on Standard Smart Cards. In: *ICT Systems Security and Privacy Protection* [online]. Springer Nature Switzerland, 2019, s. 1-13 [cit. 2020-10-16]. ISBN: 978-3-030-22312-0. Dostupné z `http://link.springer.com/10.1007/978-3-030-22312-0_20`

[7] BRODA, Jan. *Aplikace pro programovatelné čipové karty*. Brno, 2020, 46 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce doc. Ing. Jan Hajný, Ph.D.

[8] HLINKA, Jan. *Kryptografie na výkonově omezených zařízeních*. Brno, 2020, 40 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce doc. Ing. Jan Hajný, Ph.D.

[9] MORAVANSKÝ, Michal. *Implementace kryptografických protokolů na čipové karty*. Brno, 2020, 64 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Petr Dzurenda, Ph.D.

[10] *Raspberry Pi 3 Model B+* [online]. [cit. 2020-10-22]. Dostupné z: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/`

[11] *Raspberry Pi OS* [online]. [cit. 2020-10-22]. Dostupné z: `https://www.raspberrypi.org/downloads/raspberry-pi-os/`

[12] ISO/IEC 7816-3, 2006. *Identification cards – Integrated circuit cards: Part 3: Cards with contacts – Electrical interface and transmission protocols.* Third edition. Switzerland: ISO/IEC.

[13] *Node.js* [online]. [cit. 2020-11-30]. Dostupné z: `https://nodejs.org/en/`

[14] *Express* [online]. [cit. 2020-11-30]. Dostupné z: `https://expressjs.com/`

[15] Smart Card Browser Extension. *GitHub* [online]. [cit. 2021-5-2]. Dostupné z: `https://github.com/cardid/webcard`

[16] W3.CSS Tutorial. *W3Schools Online Web Tutorials* [online]. [cit. 2020-11-30]. Dostupné z: `https://www.w3schools.com/w3css/default.asp`

[17] Client-Server Overview. *MDN Web Docs* [online]. [cit. 2020-11-30]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview`

[18] WebSocket, 2001-. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 20.5.2019 [cit. 2021-5-4]. Dostupné z: `https://en.wikipedia.org/wiki/WebSocket`

[19] *MongoDB* [online]. [cit. 2021-5-5]. Dostupné z: `https://www.mongodb.com`

[20] Cron, 2001-. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2021-5-6]. Dostupné z: `https://en.wikipedia.org/wiki/Cron`

[21] *Passport.js* [online], 2014. Github [cit. 2021-5-8]. Dostupné z: `https://github.com/jaredhanson/passport`

Seznam symbolů, veličin a zkratek

APDU Application protocol data unit – zprávy pro komunikaci s čipovými katami

GDPR General Data Protection Regulation

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JSON JavaScript Object Notation

MAC Message Authentication Code

NPM Node package manager

RKVAC Revocable Keyed-Verification Anonymous Credentials

SHA Secure Hash Algorithm

URL Uniform Resource Locator

wBB Weak Boneh-Boyen signature

A Installation guide

This chapter serves as a guide for installing web applications to the system. Recommended operating system for running applications is Ubuntu Server LTS. The installation process is same for all three applications except the debugging of RKVAC application.

A.1 Node.js

Web applications are build on JavaScript back-end framework Node.js. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. The best way to install Node.js and all application dependencies is via Node Package Manager (NPM). There are multiple ways to install NPM to your system. Two easiest ways are included in this guide.

Important: Node.js version **16.x** is not supported. It is recommended to install the latest release of version **15.x** as shown in this guide.

A.1.1 Installing Node.js using NodeSource repository

1. Connect to your server's terminal and write these commands:

```
# Using Ubuntu
$ curl -fsSL https://deb.nodesource.com/setup_15.x \
$ | sudo -E bash -
$ sudo apt-get install -y nodejs
//
# Using Debian, as root
$ curl -fsSL https://deb.nodesource.com/setup_15.x \
$ | bash -
$ apt-get install -y nodejs
```

2. Following commands should return similar output:

```
$ node -v
v15.13.0
$ npm -v
7.7.6
```

A.1.2 Installing Node.js using Ubuntu official repository

1. Connect to your server's terminal and write these commands:

```
$ sudo apt-get update
$ sudo apt-get install nodejs
$ sudo apt-get install npm
```

2. Following commands should return similar output:

```
$ node -v
v15.13.0
$ npm -v
7.7.6
```

After following these instructions Node.js with Node Package Manager should be installed in the system.

A.2 Web application

After installing Node.js, download the web application to the server. Following steps shows the installation process for one of the web applications.

1. Download the web application to the server.
2. Install all dependencies for RKVAC application.
3. Follow the instructions for debugging RKVAC application with these specialties:
   - there is no need to patch `pcsc` library, due to the remote communication with the card,
   - build option for remote communication with ID card should be set to:
     `-DRKVAC_PROTOCOL_REMOTE`
   - TCP port for communication between RKVAC and web application needs to be set in the header file accordingly to web application:
     - Verifier' application – port `5000`
     - RA's application – port `5001`
     - Issuer's application – port `5002`

     The port can be specified before debugging in the header file, see Listing A.1
4. Copy the RKVAC executable to the parent folder of web application.
   - The executable should be named `rkvac-protocol-multos-1.0.0`.
5. Run application using command:

```
$ npm run serverstart
```

Výpis A.1: rkvac-server/service/config/network-config.h

```
#ifndef __RKVAC_PROTOCOL_SERVICE_NETWORK_CONFIG_H_
#define __RKVAC_PROTOCOL_SERVICE_NETWORK_CONFIG_H_

#ifdef __cplusplus
extern "C"
{
#endif

#define SRV_IPV4_ADDRESS "127.0.0.1"
#define SRV_PORT <PORT>

#define MAX_TRANSMIT 4096

#define SA struct sockaddr

#ifdef __cplusplus
}
#endif

#endif /* __RKVAC_PROTOCOL_SERVICE_NETWORK_CONFIG_H_ */
```

6. Connect to web server using address `https://<server-address>:<port>/` where:
   - **server-address** is IP address or domain name of the machine where the web app is running
   - **port** is web port of https server:
     - Verifier' application – port `8443`
     - RA's application – port `9443`
     - Issuer's application – port `10443`

B User guide

This chapter is divided in three sections, where each contains user manual for one of three applications created in the diploma thesis. For all functions to work properly, all steps stated in chapter A needs to be implemented.

B.1 Issuer' app

After connecting to the https server on address `https://<server-address>:10443/`, you will be automatically redirected to login page. Default logging-in credentials:

- username – **admin**
- password – **Vut2021**

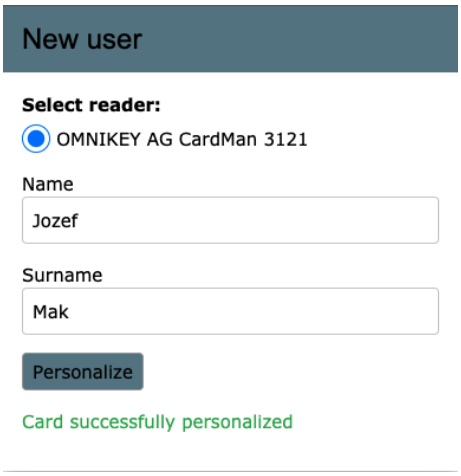
After log-in, you will be redirected to the home page.

B.1.1 Card personalization

Functions for card personalization are available on the „Users“ page. Navigation to this page is possible using both menu bar or the „Users“ panel on the home page.

On the left side of the „Users“ page a list of all personalized ID cards is placed. Each ID has a **name**, **surname** and a **ID number**, which is assigned automatically. The user list can be refreshed using a refresh button or by refreshing the web page.

For personalization of a new card connected to your PC, use the panel „New user“ placed on the right side of the page. Choose the **smart-card reader** you want to use and enter card user's **name** and **surname**. After clicking on the button „Register“ the personalization process is started. At the end of the process the success or error message is shown on the bottom of the panel (see Fig. B.1). After



Obr. B.1: Card personalization

successful personalization of a smart-card, the new entry is added to user list. For further usage a revocation handler needs to be assigned to the card (see section B.2.1).

B.1.2 Assigning attributes

Functions for assigning attributes to smart-card are available on the „credentials“ page. Navigation to this page is possible using both menu bar or the „Credentials“ panel on the home page. For assigning attributes to the given smart-card, the card needs to have a **revocation handler** assigned (see section B.2.1), and the **public key of revocation authority** needs to be uploaded to the issuer's application (see B.1.3).

Attributes are assigned to the card via credentials files. New credentials file can be created using left panel „New credentials“. Following 4 types of credentials are available:

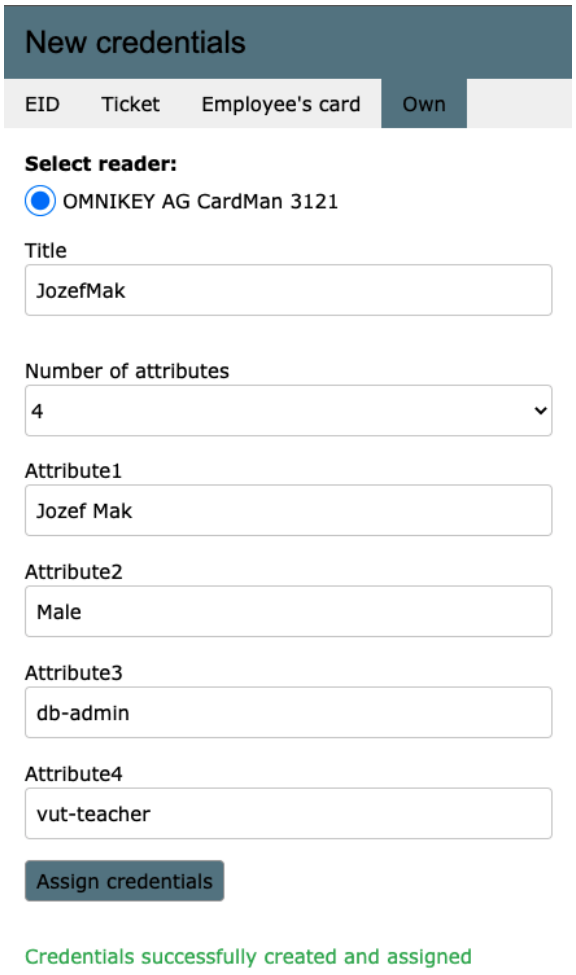
- **EID**,
- **Ticket**,
- **Employee Card**,
- **Own-defined**.

After choosing the type of credentials, appropriate attributes can be defined. In case of own-defined credentials as much as 9 attributes can be defined. After entering all attributes and clicking on „Assign credentials“ button, the new credentials file containing defined attributes is created and assigned to smart-card in chosen smart-card reader (see Fig. B.2). In the table on the left side of the page all credentials files are listed. Using appropriate icons, credentials files can be deleted, viewed and re-assigned to connected card.

B.1.3 Managing keys

For proper working of RKVAC system, the **revocation authority's public key** needs to be uploaded to issuer's application. This can be done using panel „Tools“, available on the home page of application. After choosing revocation authority's public key file from user's file system, and clicking on the „Upload“ button, the file is uploaded to the system (for obtaining the key as a revocation authority, see section B.2.2). In case of migrating RA's system, the public key can be deleted and replaced with a new one. For proper working of the application, the file containing RA's public key needs to be named `ra_pk.dat`.

Functions for managing **issuer's private key** are also available trough panel „Tools“. This key is automatically generated during the first assigning of attributes to the card. Using the button „Download“, the key can be downloaded for further



Obr. B.2: Assigning attributes

usage. In case of implementing issuer's entity from other system, the private key can be uploaded from user's file system. If the key is imported, it is not generated during the first assigning of attributes. For proper working of the application, the file containing issuer's private key needs to be named `ie_sk.dat`.

B.1.4 Smart-card readers

At the home page of the application on the panel „Smart-card Readers“, all smart-card readers plugged in the user's computer are listed. After clicking on the particular reader, the card inserted in it is contacted with SELECT AID message. In case of ACK response, the card is marked as properly working.

B.1.5 Reseting application

For resetting the application to its default state, the button on the panel „Application's reset“ can be used. Note that this will remove all RKVAC files (with exception of RKVAC executable) and any previous actions will be lost.

B.2 Revocation authority's app

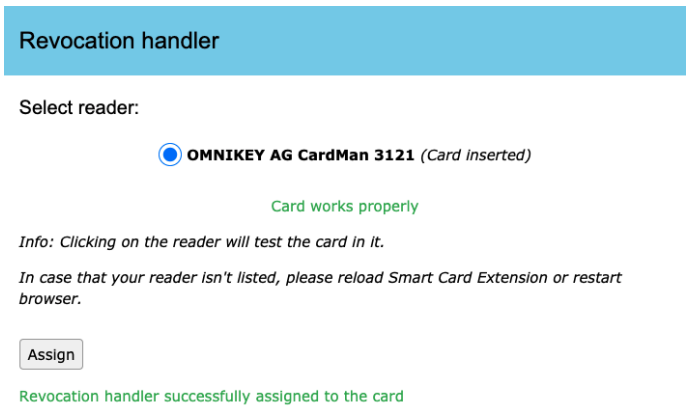
After connecting to the https server on address `https://<server-address>:9443/`, you will be automatically redirected to login page. Default log-in credentials:

- username – **admin**
- password – **Vut2021**

After logging-in, you will be redirected to the home page. In case of first usage of the application, you will be prompted to initialize the RKVAC application. After clicking on the „Initialize“ button, the appropriate folders are created on the server.

B.2.1 Revocation handler

Revocation handlers can be assigned using the panel „Assigning revocation handler“. All smart-card readers plugged in user's computer are listed in this panel. After selecting one, the card inserted in it is contacted with SELECT AID message. In case of ACK response, the card is marked as properly working. By using the „Assign“ button, the assigning process is started. On the end of the process, the appropriate message is showed on the bottom of the panel (see Fig. B.3).



Obr. B.3: Assigning revocation handler

B.2.2 Managing files

Revocation authority's **private** and **public files** can be managed using functions on the „Revocation authority's files“ panel. All these files are generated during the first usage of „Assigning revocation handler“ function. Using „Download“ buttons, files are downloaded for further usage. In case of implementing revocation authority from other system, these files can be uploaded from the local file system. Files, that are uploaded before the first usage, won't be automatically generated.

B.2.3 List of verifiers

In order to change a epoch, verifiers are connecting to revocation authority's application. RA activate their epoch, update the user blacklist and send it back to verifiers. By default revocation authority accepts connections only from IP addresses listed in „Permitted verifiers“. Functions for managing this list are placed in „Verifiers“ page, accessible through top menu bar.

Verifier's IP address can be added to the list, and then deleted. Application uses only IPv4 addresses.

B.2.4 Revoking user

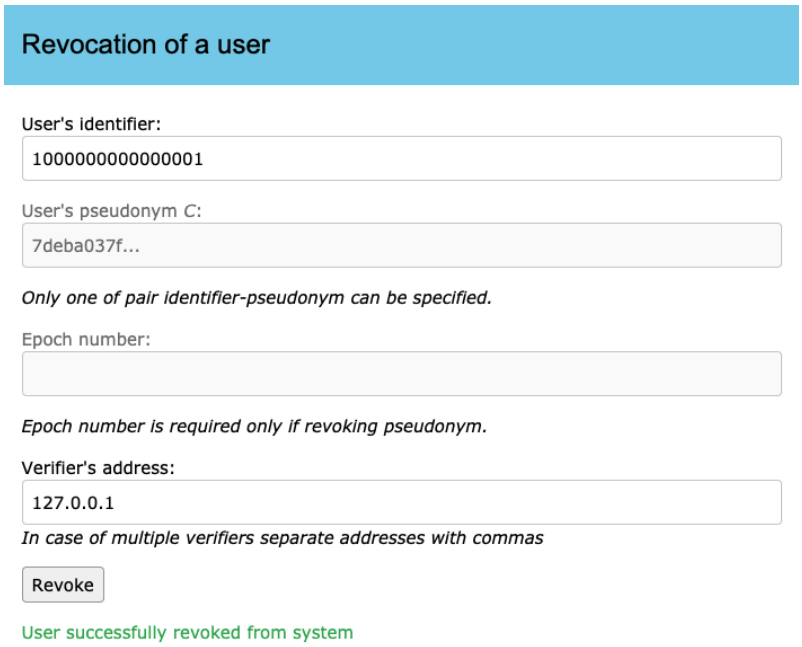
Revocation of the users from the RKVAC system, can be accomplished using functions on the „User's revocation“ panel. For a successful revocation, following information needs to be filled in the form:

- **User's ID or pseudonym** C – user's ID can be provided by an issuer (see Fig. B.1.1). Pseudonym C is available in the access logs of the verifier' app (see B.3.5).
- **Epoch number** – only if revoking pseudonym C. The number of the current verifier's epoch can be found in the verifier's app (see B.3.5).
- **Verifier's server address** – IP address or domain name of the verifier's server. In case more verifiers, addresses needs to be separated with **,**.

After providing all necessary information and starting the process by clicking on the „Revoke“ button, the blacklist for the given epoch is updated and sent to the verifier. On the verifier's side, the blacklist is automatically rewritten based on the received one, and the user is revoked from the system (see Fig. B.4).

B.2.5 Reseting application

For resetting the application to its default state, the button on the panel „Application's reset“ can be used. Note that this will remove all RKVAC files (with exception of RKVAC executable) and any previous actions will be lost.



Obr. B.4: Revocation of a user

B.3 Verifier's app

The web application that implements the verifier's site of RKVAC system, created in this diploma thesis, consists of two parts. This manual will provide steps for using the main part, which is the authentication module based on RKVAC system. The second part is the database of university classes, teachers and departments, which is just a example of a web service, that can be protected by the authentication module.

After connecting to the https server on address `https://<server-address>:8443/`, you will be automatically redirected to a login page. The login page offers two kinds of authentication:

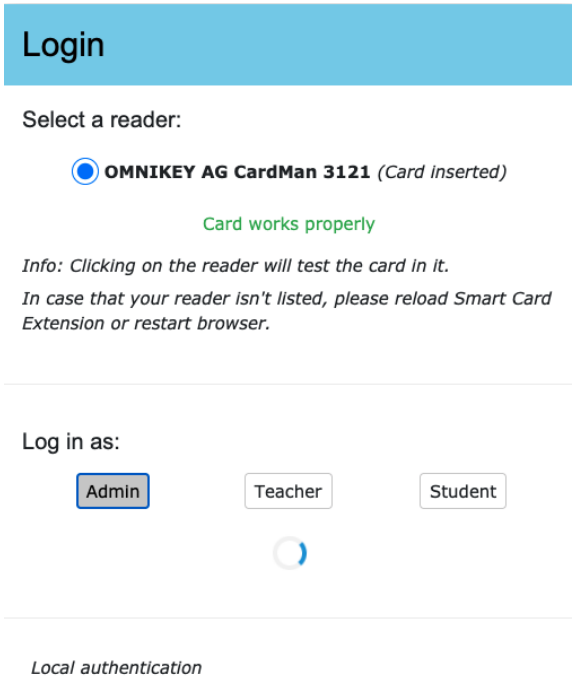
1. **RKVAC authentication** – authentication using RKVAC attributes is executed in two steps:
   (a) **Selecting the smart-card reader** – from the list of all smart-card reader, plugged in the user's machine. When a smard-card reader is selected, the card in it is tested with SELECT AID message. In case of ACK answer, the smart-card is marked as properly working.
   (b) **Choosing the user-role** – after selecting the smart-card reader, user needs to select the user-role, with which he wants to authenticate towards the application. Following three user-roles are available – **Admin**, **Teacher**, **Student**.

   When a user-role is selected, the authentication process automatically starts (see Fig. B.5). User's smart-card is contacted with challenge to provide the

proof of holding the required attributes. In case of successful verification of the proof, user is logged in, and is authorized for actions based on the selected user-role. In case of authentication failure the appropriate message is shown.

2. **Local authentication** – At the default state, the application is not ready for RKVAC authentication. For administration purposes the local authentication is placed in the bottom of the login page. Default log-in credentials:
   - username – **admin**
   - password – **Vut2021**

   After logging-in, you will be redirected to the home page.



Obr. B.5: RKVAC authentication

Whole RKVAC administration is based at the „Setup“ page, which is available through the left-side panel. Along with „Access logs“ page (see B.3.5), only administrators are permitted to access this page.

In case of first usage of the application, you will be prompted to initialize the RKVAC application. After clicking on the „Initialize“ button, the appropriate folders are created on the server.

B.3.1 Managing keys

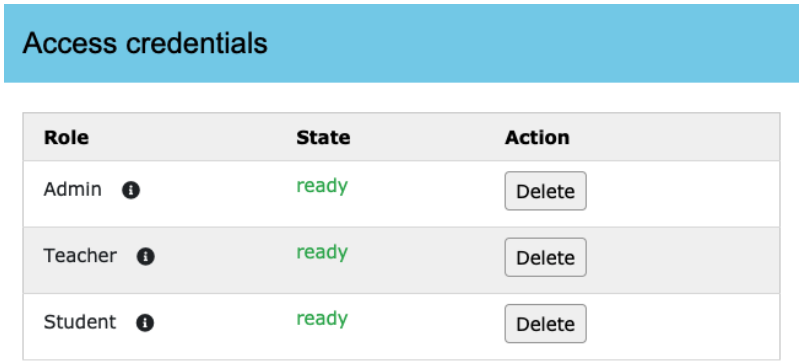
For proper working of RKVAC system, verifier's application needs to have access to following files:

- **issuer's private key**
- **revocation authority's public key**
- **revocation authority's public parameters**

All these files can be uploaded via the „RKVAC keys" panel. In case of system migration, all files can be later deleted and replaced with other ones.

B.3.2 Access credentials

Functions for administration of access credentials are placed in „Access credentials" panel. In the initiate state of the app, none of the access credentials are created. As an admin we need to prepare credentials for each user-role, that will have an access to the system. In the credentials we define attributes, that we want to verify, and their position. The current state of the access credentials is displayed in the table (see Fig. B.6). If the access credentials for one of the user-roles are created, they are marked as **ready**, and there is a possibility to delete them. Details of the credentials



Obr. B.6: Access credentials table

can be displayed via „info" icon, placed next to the user-role name.

Creation of a new access credentials is performed in these steps:

1. **Choosing user-role** – each role is available for selecting, only if it doesn't have any other credentials prepared.
2. **Selecting attributes count** – number of attributes in the credentials needs to be the same, as the number of attributes in the credentials, with which the users of RKVAC system would request to log-in to the application.
3. **Defining attributes** – only the attributes that will be used for logging-in needs to be declared. Other filed can be left blank. Attributes needs to be set up on the exact same position, as are the same attributes in the user's credentials.
4. **Required attributes position** – parameter specifies the position of attributes, that will be requested during the authentication process.

If the user has assigned the credentials with these attributes:

- attribute 1: **Jozef Mak**
- attribute 2: **Male**
- attribute 3: **db-admin**
- attribute 4: **vut-teacher**

then the credentials that would provide him admin access to this database could have this structure:

- attribute 1:
- attribute 2:
- attribute 3: **db-admin**
- attribute 4: **vut-teacher**
- Required attributes position: **3,4**

With this example set as a admin credentials, any user that provides prove of holding attributes **db-admin** and **vut-teacher** in the positions **3** and **4** in his credentials, would be logged-in with administrator access.

B.3.3 Epoch management

Functions for epoch management are placed in the panel „Epoch settings“. At the top of this panel, a current epoch number is stated. After using the button „Switch now“ a new epoch is generated, and sent to the revocation authority for activation.

For proper working of the epoch switching function, an **address of revocation authority's server** needs to be set up. After entering the IP address or domain name of the server, it is saved in the back-end of the application, and the epoch switching function is enabled.

At the bottom of the „Epoch settings“ panel, function for scheduling a regular epoch switching is available. Time expression needs to be entered in the **crontab format**. The expression consists of 5 fields separated by space. Each field defines different time unit of the scheduling job – „`<minute> <hour> <day(month)> <month> <day(week)>` “, where the character `*` defines the value „each“. Practically an expression `0 0 * * *` would schedule the task to be run each day at 0:00 every month. Another example is an expression `45 * * * *`, which would schedule the task to be run every hour at 45^{th} minute. Admin can use the crontab help, available at following link `https://crontab.guru`.

B.3.4 Reseting application

For resetting the application to its default state, the button on the panel „Application's reset“ can be used. Note that this will remove all RKVAC files (with exception of RKVAC executable) and any previous actions will be lost.

B.3.5 Access logs

User with administrator privileges can access the web page „Access logs“, available through left-side panel. At this page, the list of 50 most recent access logs are available. Each log consists of:

- **timestamp,**
- **epoch number,**
- **pseudonym** C**,**
- **result** – access **ALLOWED** or **DENIED**.

User's pseudonym C and epoch number, can be sent to the revocation authority in order to revoke the user from RKVAC system.

Logs are displayed in reverse order – the most recent are on the top of the list. The list is limited to contain the last 50 records. Older logs can be found directly on the server in the file `./data/Verifier/ve_requests.log`.

C Proof of concept

In this chapter a full proof of concept is described. Following these steps, you will test the majority of the functions of web applications, created in this diploma thesis. All steps described in this chapter are for demonstration purpose. For easier start, it is recommended to use prepared virtual machines for both server and client appliances, that are saved in the shared folder on Google Drive.
The folder is available for all members of the group „Brno University of Technology“ and it contains two virtual machines:

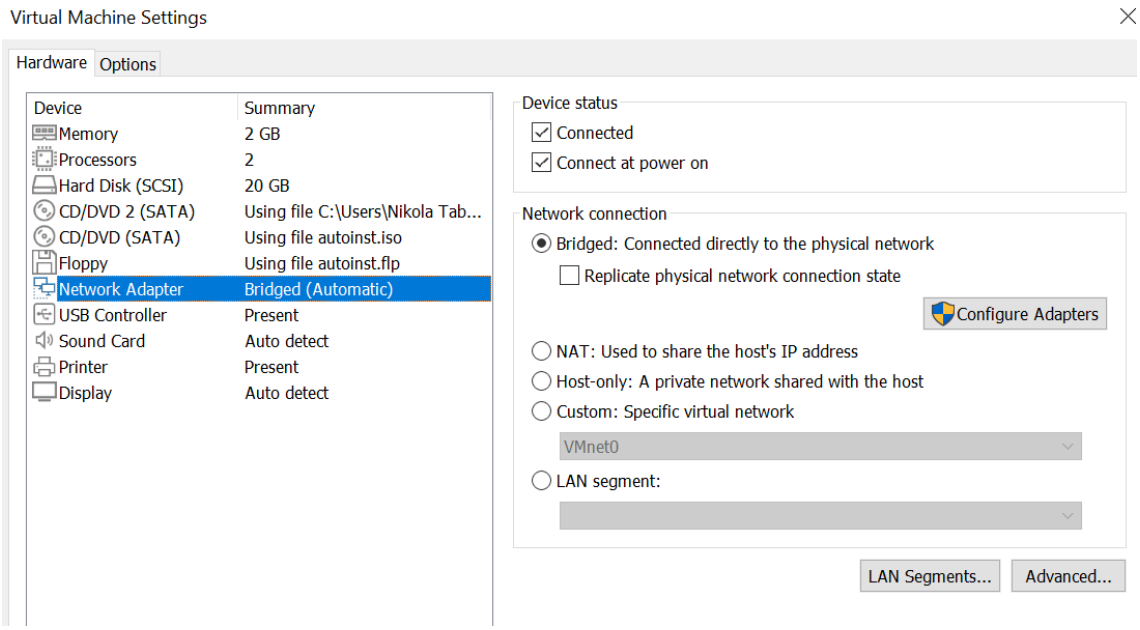
- **RKVAC Server** – Ubuntu Server 20.04 LTS. This appliance serves as a server, on which all three web applications are running. The following structure is prepared on the machine:
  - all dependencies of RKVAC application are installed,
  - the RKVAC application for each web application is compiled and ready,
  - all dependencies from „Installation guide“ are installed,
  - each RKVAC entity has its own user account, created for simulation of independent environments,
  - in `/home` folder of each user, there is a directory named `rkvac-<entity>`, which contains cloned GitHub repository of the particular application.
- **RKVAC Client** – Windows Desktop 10. This appliance serves as a client for all web applications. The only thing that is prepared in this appliance is installed Google Chrome with the Smart Card Extension.

C.1 Server preparation

In this section you will prepare the server. At the end you will have all applications running on the virtual machine.

1. Download the virtual machine **rkvac-server** from the link in the introduction of this chapter.
2. Open the virtual machine in VMware Workstation.
3. Before starting the machine, make sure that the network adapter is set in the mode **bridged** – see Fig. C.1. In this mode the virtual machine will be connected directly to your local network.
4. Start the virtual machine.

After booting the virtual machine you should see on login page addresses of web applications (see Fig. C.2). All application should be running on specified port on IP address of the machine.



Obr. C.1: Network adapter needs to be set to **bridged**

Obr. C.2: Server on startup

C.2 Client preparation

Client is a Windows Desktop virtual machine, that have installed Google Chrome with Smart Card Extension. In these few steps you will prepare the client's machine, so that you will be able to use RKVAC functions in the application.

1. Download virtual machine rkvac-client
2. Make sure that the network adapter of the virtual machine is set to „bridged“ (see Fig. C.1) – we won't need client's IP address in order to use web applications.
3. Start the virtual machine.
4. Log in using credentials:
   - username – **admin**

- password – **Vut2021**

5. Connect a smart card reader with appropriate card to your computer.
6. Connect the reader to the virtual machine – this is available through „Virtual Machine Settings –> USB Controller“
7. Start Google Chrome

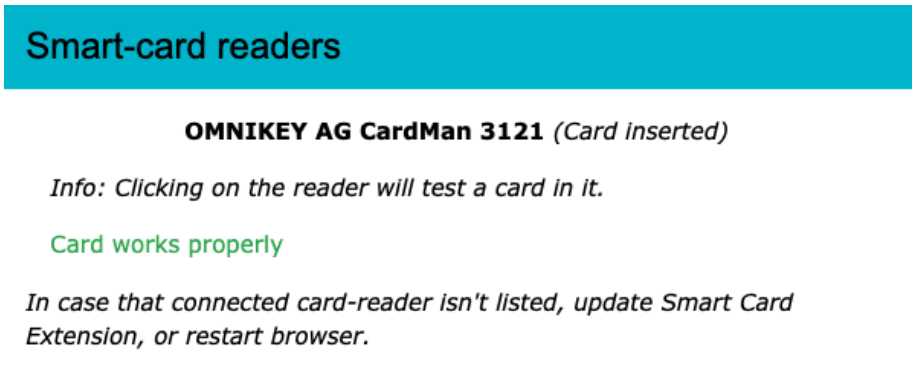
C.3 Assigning attributes to card

In this section the process of integration of the card to the system is described. In order to fully integrate the card you need to:

- **personalize the card,**
- **assign revocation handler to the card,**
- **import revocation authority' key to the issuer,**
- **assign attributes to the card.**

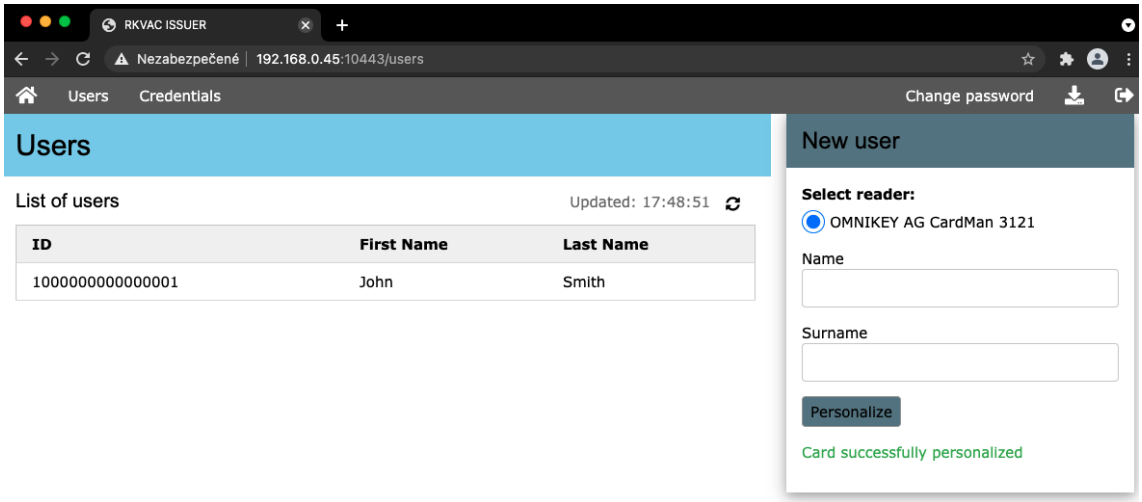
C.3.1 Personalization of the card

1. Connect to issuer's app on the address:
   - `https://<server-address>:10443`
2. Log in using credentials:
   - username – **admin**
   - password – **Vut2021**
3. Test the smart-card – on the home page you should see your smart-card reader listed on the panel „Smart-card readers“ (see Fig. C.3). By clicking on the reader, application will test the inserted card with `APDU_SELECT` message.
   Info: if you don't see your reader listed, please update Smart Card Extension or restart the browser.



Obr. C.3: Testing smart card

4. Navigate to page „Users“ using the menu bar on the top.
5. On the form „New User“ choose your smart-card reader and enter:
   - first name: **John**
   - last name: **Smith**
6. Click „Register“
7. If you see the success message and the new user is listed in the „User list“, the card personalization was successful (see Fig. C.4).



Obr. C.4: Card's personalization

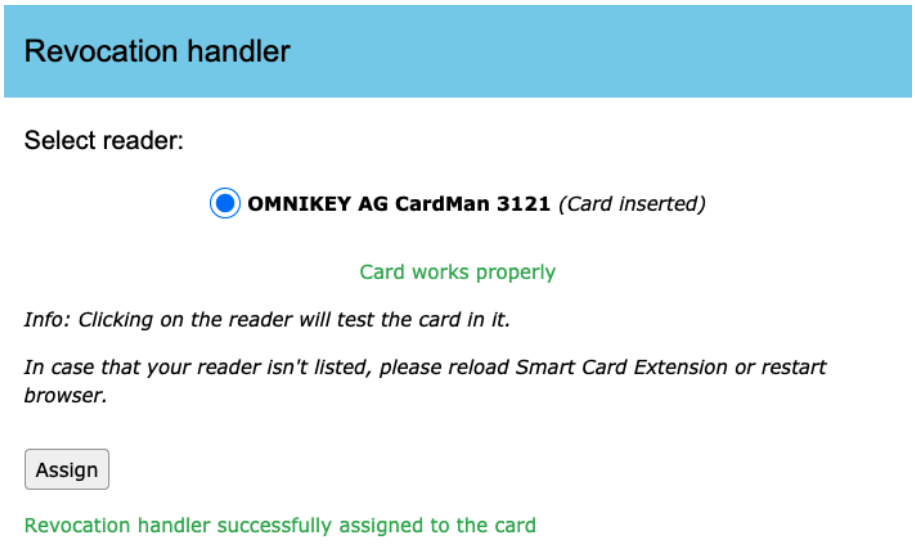
C.3.2 Assigning revocation handler

After personalization of the card, you need to assign a revocation handler to it.

1. Connect to revocation authority's app on address:
   - `https://<server-address>:9443`
2. Log in using credentials:
   - username – **admin**
   - password – **Vut2021**
3. Initiate the app using the initiate button.
4. Select your smart-card reader
5. Assign revocation handler to the card.
6. If you see the success message, the process was successful (see Fig. C.5).

C.3.3 Permitting verifier

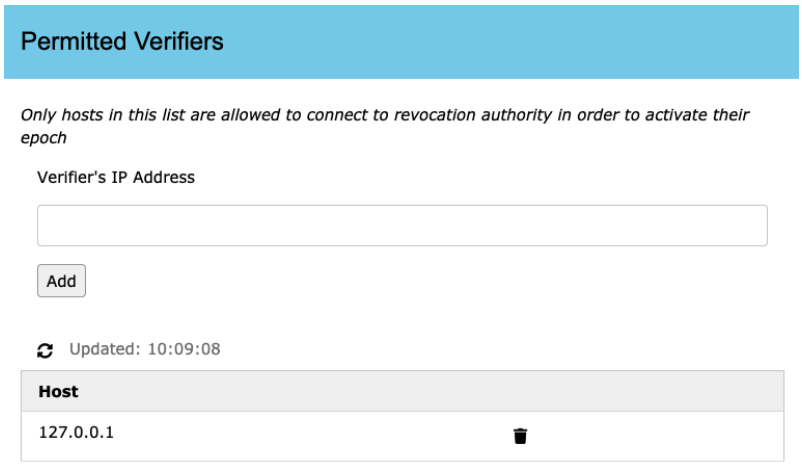
In order to allow connection from the verifier's app, you need to add his IP address to „Permitted verifiers“ list. Verifier will need to connect to RA's app during switching to new epoch (see C.4.3).



Obr. C.5: Assigning revocation handler

1. Navigate to „Verifiers“ page using menu bar.
2. Add address **127.0.0.1** – verifier's application runs on a same machine, so it will use the localhost address.

List of permitted verifiers should looks like Fig. C.6.



Obr. C.6: Permitted verifiers

C.3.4 Managing the keys

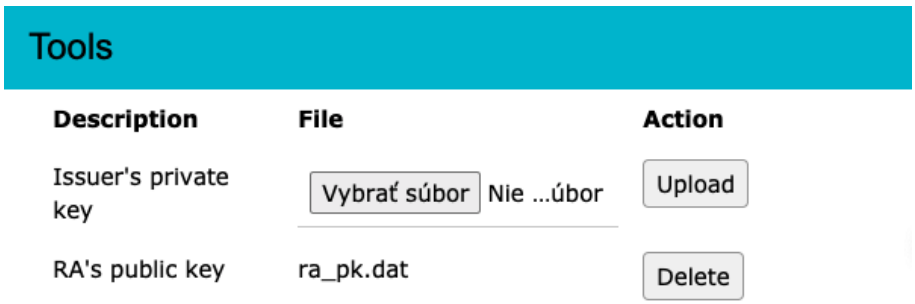
During the first usage of the revocation authority's app, private and public keys were generated. In order to use the issuer's and verifier's app, you need to integrate some of these keys to their system.

1. Download `ra_pk.dat` and `ra_public_parameters` from panel „Revocation authority's files“.

Note: please be careful if you already have some other files in your `Downloads/` folder with the same name. You need to delete the old files, so that these actual files have the appropriate names.

3. Log into the issuer's app – you were probably log-out from this app, so you need to **refresh the page** and re-log in.
4. Import the `ra_pk.dat` to the issuer's system using „Tools“ panel.

After importing the appropriate file, you should see that the card „Credentials“ was turn to blue (see Fig. C.7). This indicates that you have all dependencies set up and you can assign attributes to the card.

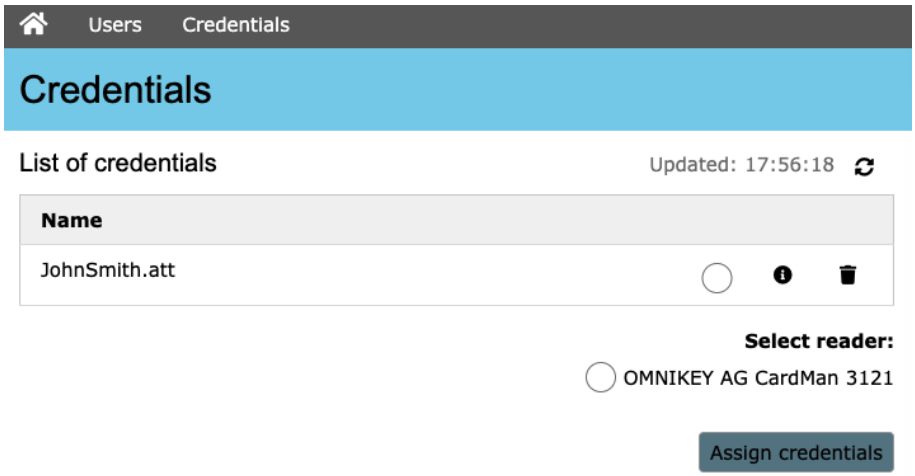


Obr. C.7: Importing RA's keys

C.3.5 Assigning attributes

Using this example, the card will have assigned attributes, which will provide access to the verifier's system with role **teacher**.

1. Navigate to the „Credentials“ panel.
2. On the panel „New credentials“ choose „Own“.
3. Select your smart-card reader.
4. In the form enter these values:
   - Name – **JohnSmith**
   - Attribute count – **4**
   - Attribute 1 – **John Smith**
   - Attribute 2 – **Male**
   - Attribute 3 – **vut-teacher**
   - Attribute 4 – **db-teacher**
5. Click „Assign“.
6. If you see the success message and you can see the file `JohnSmith.att` in the credentials list, the process was successful (see Fig. C.8).



Obr. C.8: Created new credentials file

C.4 Verifying attributes

In this section the verifier's app will be set up to grant the teacher's access for user's that can proof, that hold attributes **vut-teacher** and **db-teacher** on the positions **3** and **4** of theirs credentials.

C.4.1 Integrating verifier to RKVAC

As a first step, you need to integrate the verifier to the RKVAC system. This can be done by importing revocation authority's public files and issuer's private key into the verifier's system.

1. In issuer's app, navigate to the home page.
2. Download `ie_sk.dat`.

You should have the revocation authority's public files downloaded from earlier. If not, please see subsection C.3.4.

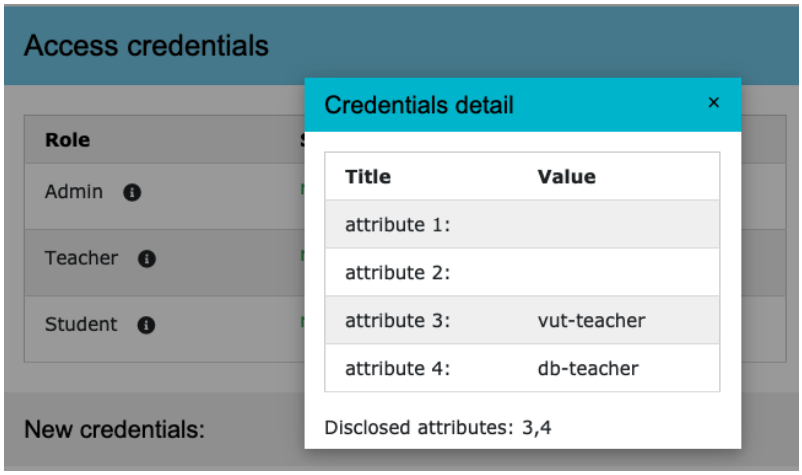
4. Connect to verifier's app at the address:
   - `https://<server-address>:8443`
5. Log in using local authentication and credentials:
   - username – **admin**
   - password – **Vut2021**
6. Navigate to „Settings“ using the left panel.
7. Initiate the app using the appropriate button.
8. Import the downloaded files to the system using „RKVAC keys“ panel.

C.4.2 Preparing access credentials

Using these steps you will prepare access credentials for teacher's access. Similarly you can later prepare credentials for both admins and students.

1. On the panel „New credentials“ choose „Teacher“.
2. In the form enter these values:
   - Attributes count – **4**
   - Attribute 1 – leave empty
   - Attribute 2 – leave empty
   - Attribute 3 – **vut-teacher**
   - Attribute 4 – **db-teacher**
   - Required attributes position – **3,4**
3. Click „Create“

In the result the prepared teacher's credentials should look like Fig. C.9. You can check credentials details through the table above, where this credentials should be marked as **ready**.



Obr. C.9: Teacher's access credentials

C.4.3 Connecting to RA

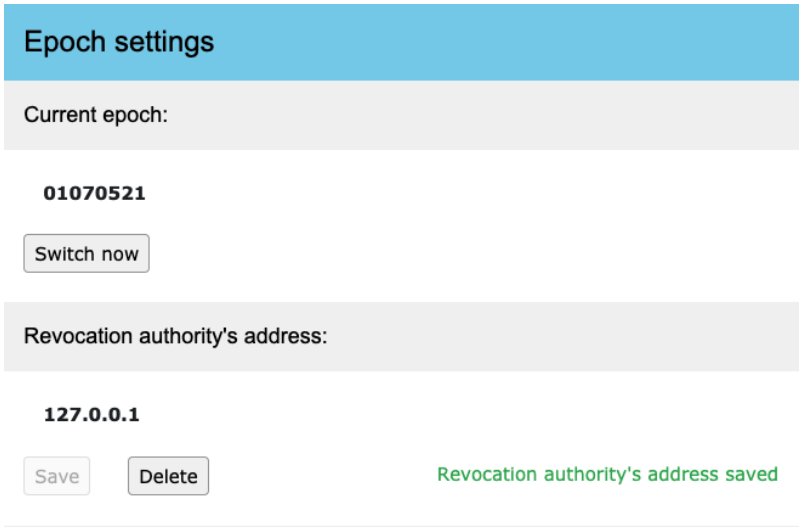
In this subsection you will finalize verifier's settings by connecting it to revocation authority's server.

1. In the panel „Epoch settings“ in the revocation authority's address field enter **127.0.0.1**
2. Click „Save“
3. Click „Change now“

After following these steps, the revocation authority's address (in this case localhost) is saved to the verifier's system and a new epoch was successfully generated (see Fig.

C.10).

At this point you have everything set up for successful authentication. Log out



Obr. C.10: Epoch settings

from the verifier's app and on the login page choose your smart-card reader and click on „Teacher“ button. This will start the verification process. If everything was correctly set up, you should be logged in the app with teacher's access.

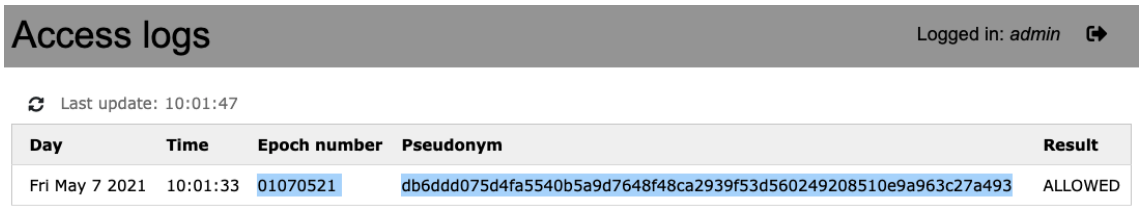
C.5 Revoking user

In this section you will revoke the card from the RKVAC system. There are two possibilities on how to revoke user:

- using **user's ID**
- using **user's pseudonym** C and **epoch number**

Using these steps, you will revoke user using his **pseudonym** C and current epoch number.

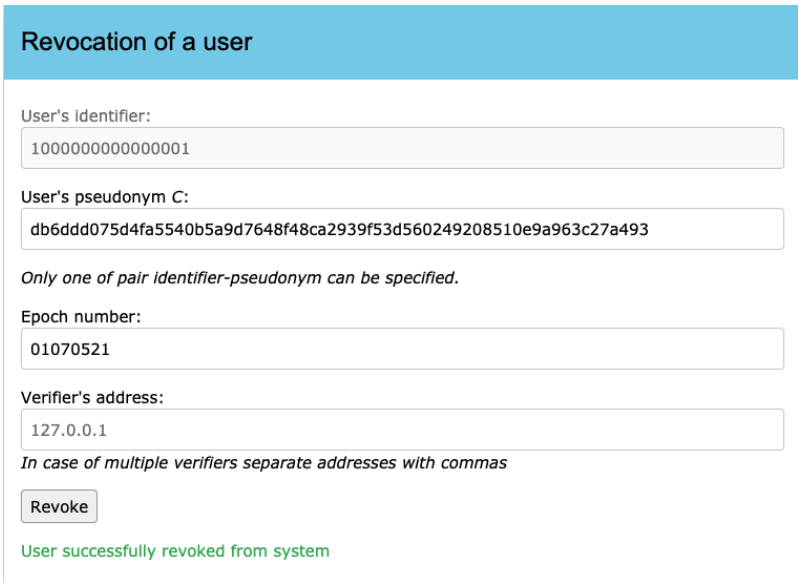
1. Log into the verifier's application as Admin (using local authentication).
2. Navigate to „Access logs“ page using left menu bar.
3. Write down **Epoch number** and **Pseudonym** from the last log (see Fig. C.11).
4. Log into the revocation authority's app.
5. On panel „Revocation of user“ enter:
   - **User's ID** – leave empty.
   - **Pseudonym** C – user's pseudonym from access log.



Obr. C.11: Access log

- **Epoch number** – epoch number from access log.
- **Verifier's address** – leave empty – if left empty, localhost address is automatically used.

6. Click „Revoke".
7. If the revocation was successful you should see an appropriate message (see Fig. C.12).



Obr. C.12: Revocation of a user

At this point your card should be revoked from the system. Connect to verifier's app and try to log in as „Teacher", using your smart card. At the end of the verification process, you should see message „Access denied". In case you want to add this card to the system, you need to follow all steps from section C.3.

D Protokol z testování

Táto zpráva shrnuje výsledky z testování webových aplikací, vytvořených v diplomové práci. Výkonnostně nejvíc náročné jsou operace provádějící jednotlivé protokoly schématu RKVAC a proto bylo testování zaměřené na tyto operace. RKVAC protokoly vykonává RKVAC aplikace, která je implementovaná v rámci každé webové aplikace.

D.1 Způsob měření

Měření trvání jednotlivých operací bylo provedeno v dvou intervalech.

- První interval je časový okamžik od odeslání požadavku klientem, po přijetí odpovědi. Jedná se o reálnou časovou náročnost, která se projeví během používání aplikace. Aplikace byla při testování nasazena na virtuálním serveru, který se nachází v České republice.
- Druhý interval je časový okamžik od spuštění RKVAC protokolu pomocí RKVAC aplikace, po vrácení výsledku. Tato doba je měřená výhradně na back-endu serveru a jedná se o náročnost aplikace RKVAC, která je v rámci webových aplikací implementována.

Třetí hodnota v každém měření ukazuje rozdíl mezi těmito časy. Tento údaj nám ukazuje čas, který trvá webový aplikaci práce na požadavku, spolu se zasláním HTTPS odpovědi klientovi. Jedná se o nejvýřečnější údaj o výkonnosti webové aplikace.

D.2 Výsledky měření

V této sekci jsou uvedeny výsledky měření jednotlivých operací. Každá operace byla měřena 5 krát a výsledky byly zprůměrovány.

D.2.1 Personalizace karty

Číslo měření	**1**	**2**	**3**	**4**	**5**	**Průměr**
Celkový čas [ms]	1412	1426	1415	1402	1420	**1415**
Back-end čas [ms]	1282	1284	1283	1272	1291	**1282,4**
Aplikačný čas [ms]	130	142	132	130	129	**132,6**

Tab. D.1: Výsledky testů operace „Personalizace karty“

D.2.2 Vydání atributů

Číslo měření	1	2	3	4	5	Průměr
Celkový čas [ms]	1931	1903	1898	1910	1887	**1905,8**
Back-end čas [ms]	1789	1771	1757	1772	1756	**1769**
Aplikačný čas [ms]	142	132	141	138	131	**136,8**

Tab. D.2: Výsledky testů operace „Vydání atributů“

D.2.3 Vydání revokačního handleru

Číslo měření	1	2	3	4	5	Průměr
Celkový čas [ms]	3284	2175	2151	2228	2152	**2398**
Back-end čas [ms]	3174	2138	2111	2186	2114	**2344,6**
Aplikačný čas [ms]	110	37	40	42	38	**53,4**

Tab. D.3: Výsledky testů operace „Vydání revokačního handleru“

D.2.4 Ověření atributů

Číslo měření	1	2	3	4	5	Průměr
Celkový čas [ms]	5434	5425	5426	5337	5359	**5396,2**
Back-end čas [ms]	5041	5086	5108	5068	5098	**5080,2**
Aplikačný čas [ms]	393	339	318	269	261	**316**

Tab. D.4: Výsledky testů operace „Vydání atributů“

D.3 Zhodnocení výsledků

Z výsledků je patrné, že kapacitně nejnáročnějšími operacemi jsou protokoly schématu RKVAC. Tyto protokoly spouští RKVAC aplikace a jejich optimalizace proto není předmětem této práce.

Vrstva webových aplikací, které byly v této práci vytvořené, je kapacitně málo náročná. Rozložení zátěže mezi webovou a RKVAC aplikaci je možné vidět v Tab. D.5. Z celkové kapacity během nejnáročnějších operacích zabírá webová aplikace méně než 10 procent a je proto možné ji považovat za optimalizovanou.

Operace	RKVAC aplikace	Webová aplikace
Personalizace karty	91,6 %	9,4 %
Vydání atributů	92,83 %	7,17 %
Vydání revokačního handleru	97,8 %	2,2 %
Ověření atributů	94,14 %	5,86 %

Tab. D.5: Rozložení zátěže mezi RKVAC a webovou aplikaci

E Obsah adresáře s aplikací

K této práci jsou přiloženy tři složky se zdrojovými kódy webových aplikací. V následujícím výpisu jsou přehledně vypsány všechny podsložky a soubory. Složky, ke kterým není uveden popis, jsou prázdné, nebo se soubory v nich nepoužívají. Jedná se o připravené složky nebo soubory pro další použití v pokračující diplomové práci.

E.1 Aplikace vydavatele

- `/` — kořenový adresář aplikace vydavatele
  - `bin` — složka pro spouštěné soubory
    - `www` — počáteční bod aplikace
  - `public` — front-end aplikace
    - `images`
    - `javascripts` — skripty klienta
    - `stylesheets` — design stránek
    - `webfonts`
  - `routes` — zdrojové kódy pro routery
    - `auth.js` — router pro autentizaci
    - `index.js` — router pro hlavní část aplikace
  - `views` — HTML webové stránky
  - `README.md` — README soubor aplikace
  - `app.js` — základní nastavení aplikace
  - `package.json` — definice použitých balíčků
  - `package-lock.json` — definice použitých balíčků v verzemi
  - `passwd` — soubor s heslem
  - `server.cert` — self-signed certifikát pro HTTPS
  - `server.key` — klíč pro HTTPS

E.2 Aplikace ověřovatele

- `/` — kořenový adresář aplikace vydavatele
  - `bin` — složka pro spouštěné soubory
    - `www` — počáteční bod aplikace
  - `controllers` — ovládání databáze
  - `models` — prvky databáze
  - `public` — front-end aplikace
    - `images`
    - `javascripts` — skripty klienta
    - `stylesheets` — design stránek
    - `webfonts`
  - `routes` — zdrojové kódy pro routery
    - `auth.js` — router pro autentizaci

```
    └── index.js ..... router pro hlavní část aplikace
├── views ..... HTML webové stránky
├── README.md ..... README soubor aplikace
├── app.js ..... základní nastavení aplikace
├── key.pem ..... klíč pro HTTPS
├── package.json ..... definice použitých balíčků
├── package-lock.json ..... definice použitých balíčků v verzemi
├── passwd ..... soubor s heslem
└── server.cert ..... self-signed certifikát pro HTTPS
```

E.3 Aplikace revokační autority

```
/ ..... kořenový adresář aplikace vydavatele
├── bin ..... složka pro spouštěné soubory
│   └── www ..... počáteční bod aplikace
├── public ..... front-end aplikace
│   ├── images .....
│   ├── javascripts ..... skripty klienta
│   ├── stylesheets ..... design stránek
│   └── webfonts .....
├── routes ..... zdrojové kódy pro routery
│   ├── auth.js ..... router pro autentizaci
│   └── index.js ..... router pro hlavní část aplikace
├── views ..... HTML webové stránky
├── README.md ..... README soubor aplikace
├── app.js ..... základní nastavení aplikace
├── package.json ..... definice použitých balíčků
├── package-lock.json ..... definice použitých balíčků v verzemi
├── passwd ..... soubor s heslem
├── server.cert ..... self-signed certifikát pro HTTPS
└── server.key ..... klíč pro HTTPS
```